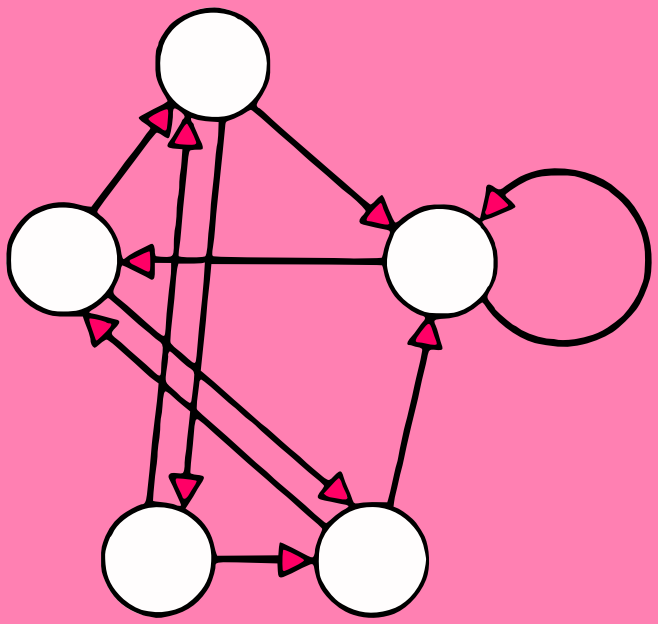
S. Baranov

SYNTHÈSE DES AUTOMATES MICRO-PROGRAMMÉS



Éditions Mir Moscou

С. И. БАРАНОВ

СИНТЕЗ МИКРОПРОГРАММНЫХ АВТОМАТОВ

ИЗДАТЕЛЬСТВО «ЭНЕРГИЯ» ЛЕНИНГРАД 1979

S. BARANOV

SYNTHÈSE DES AUTOMATES MICROPROGRAMMÉS

ÉDITIONS DE MOSCOU

Traduit du russe
par Boris PAVLOV

À NOS LECTEURS

Les Editions Mir vous seraient très reconnaissantes de bien vouloir leur communiquer votre opinion sur le contenu de ce livre, sa traduction et sa présentation, ainsi que toute autre suggestion.

Notre adresse:
2, Pervi Rijski péréoulok,
Moscou, I-110, GSP, U.R.S.S.

На французском языке

© Издательство «Энергия» Ленинград 1979
© Traduction française Editions Mir 1983

TABLE DES MATIÈRES

Extrait de l'avant-propos à la première édition russe 7
Extrait de l'avant-propos à la deuxième édition russe 9
Avant-propos à l'édition française 10

Chapitre premier. ÉLÉMENTS DE THÉORIE DES ENSEMBLES . . 13

1-1. L'ensemble 13
1-2. Opérations sur les ensembles 15
1-3. Correspondance. Fonction. Relation 17
1-4. Relations d'équivalence et d'ordre 21

Chapitre 2. AUTOMATE ABSTRAIT 25

2-1. Définition d'un automate abstrait 25
2-2. Automates de Mealy et de Moore. Les différents modes de définition d'un automate 27
2-3. Lien entre les modèles de Mealy et de Moore 36
2-4. Modèle composite (C-automate) 41

Chapitre 3. MINIMISATION DES AUTOMATES 43

3-1. Classes de compatibilité maximales 43
3-2. Recouvrement fermé minimal 50
3-3. Minimisation des automates partiels à l'aide des classes premières 53
3-4. Une méthode approchée de minimisation des automates partiels . . 62
3-5. Minimisation des automates complètement spécifiés. 65

Chapitre 4. COMPOSITION ET DÉCOMPOSITION DES AUTOMATES 70

4-1. Connexion de deux automates 70
4-2. Connexion à fonction de sortie 76
4-3. Réseau d'automates 78
4-4. Partitions sur l'ensemble d'états de l'automate résultant d'un réseau 85
4-5. Partitions à propriété de substitution 89
4-6. Couples de partitions 93
4-7. Théorème général de décomposition 95
4-8. Matrice structurelle du réseau d'automates 105

Chapitre 5. AUTOMATE STRUCTURÉ 109

5-1. Méthode canonique de synthèse structurelle des automates 109
5-2. Exemple d'application de la méthode canonique de synthèse structurelle 115
5-3. Synthèse de l'automate avec des bascules *D*, *T* et *RS* 121
5-4. Méthode graphique de synthèse structurelle des automates 126
5-5. Codage des états. Courses dans l'automate 130
5-6. Codage des états et complexité du circuit combinationnel 133

Chapitre 6. SCHÉMAS OPÉRATORIELS D'ALGORITHMES . . . 142

6-1. Microprogrammes de fonctionnement des réseaux séquentiels . . . 142
6-2. Schémas-graphes d'algorithmes 143
6-3. Schémas-graphes d'algorithmes explicités 149
6-4. Schémas logiques d'algorithmes 151
6-5. Formules de transition 158
6-6. Schémas matriciels d'algorithmes 162

Chapitre 7. TRANSFORMATION DES SCHÉMAS OPÉRATORIELS D'ALGORITHMES 164

7-1. Minimisation des sommets conditionnels sur le schéma-graphe d'algorithme 164
7-2. Utilisation de la distribution des décalages pour la minimisation des schémas opératoriels d'algorithmes 187
7-3. Minimisation du schéma-graphe d'algorithme par réduction des sommets opératoriels 190
7-4. Fusionnement des schémas-graphes d'algorithmes 195

Chapitre 8. SYNTHÈSE DES AUTOMATES MICROPROGRAMMÉS D'APRÈS LE SCHÉMA-GRAPHE D'ALGORITHME . . 208

8-1. Synthèse d'un automate microprogrammé de Mealy 208
8-2. Synthèse d'un automate microprogrammé de Moore 214
8-3. Minimisation des automates microprogrammés 218

Chapitre 9. SYNTHÈSE DU SHCÉMA LOGIQUE DE L'AUTOMATE MICROPROGRAMMÉ 223

9-1. Table de structure d'un automate microprogrammé 223
9-2. Prédécodeur de retour 226
9-3. Fonctions d'excitation complétées 228
9-4. Méthode de décomposition du schéma logique à l'aide des nœuds du schéma-graphe d'algorithme 229
9-5. Factorisation des fonctions booléennes 239
9-6. Synthèse du schéma logique d'un automate microprogrammé d'après sa table de structure avec des nœuds 253

Chapitre 10. RÉALISATION MATRICIELLE DES AUTOMATES MICROPROGRAMMÉS 258

10-1. Réalisation matricielle des réseaux combinationnels 258
10-2. Réalisation matricielle triviale d'un automate microprogrammé 262
10-3. Passage à un nouvel ensemble de variables d'entrée lors d'une réalisation matricielle de l'AMP 267

10-4. Codage des micro-instructions 276
10-5. Construction du schéma logique 282
10-6. Décomposition des matrices en réalisation triviale 285

Chapitre 11. SYNTHÈSE DES AUTOMATES MICROPROGRAMMÉS AVEC DES MATRICES LOGIQUES PROGRAMMABLES 293
11-1. Matrices logiques programmables 293
11-2. Synthèse des automates microprogrammés avec des MLP combinationnelles 298
11-3. Réalisation triviale de l'automate microprogrammé avec des MLP à mémoire 309
11-4. Décomposition de l'automate microprogrammé 311
11-5. Synthèse des automates microprogrammés avec des MLP à mémoire 320
Bibliographie 331
Index alphabétique des matières 334

A la mémoire du très regretté
professeur Victor Sélioutine,
qui fut mon collègue et ami.

EXTRAIT DE L'AVANT-PROPOS À LA PREMIÈRE ÉDITION RUSSE

La complexité sans cesse croissante des systèmes automatiques et des complexes de calcul à données discrètes d'une part, et la nécessité de satisfaire à des exigences — souvent contradictoires — de fiabilité, poids, encombrement, rapidité de fonctionnement et réalisation dans des délais acceptables d'autre part, rendent absolument indispensable le recours sur une grande échelle à la formalisation et à l'automatisation de toutes les étapes de la conception. Les méthodes intuitives basées sur l'expérience de l'ingénieur sont mises en défaut lorsqu'il s'agit de rechercher des solutions proches des optimales et ceci dans des délais très restreints. Or, l'appareil développé de la logique mathématique et de la théorie des automates pourrait bien venir en aide au chercheur. Cependant les chercheurs n'en profitent que rarement. A notre avis, la cause en est que la littérature disponible soit n'aborde que l'aspect essentiellement théorique des choses, soit ne traite que de questions d'utilité restreinte. Le but que s'est fixé l'auteur du présent ouvrage est d'essayer de combler le fossé entre la théorie et la nécessité pratique, de mettre les acquis de la théorie des automates à la portée du concepteur des systèmes numériques.

L'ouvrage expose les techniques de synthèse permettant de réaliser des automates microprogrammés complexes, à grand nombre d'entrées et de sorties et à un nombre d'états internes de l'ordre de quelques centaines. Il ne requiert aucune connaissance préalable dans ce domaine : toutes les notions et définitions indispensables à la compréhension de la théorie sont données dans les chapitres liminaires. Un grand nombre d'exemples tirés de la pratique de l'ingénieur facilitent l'assimilation de l'exposé. Chaque fois qu'il a fallu faire un choix entre le concret et la rigueur de l'exposé, l'auteur a opté pour le concret. Le cadre restreint du livre nous a obligés à sauter toute une série de questions des automates numériques et des circuits logiques. Le lecteur désireux d'en connaître plus consultera avec profit les monographies bien étoffées de M. Gavrilov, V. Glouchkov, A. Zakrevski, V. Lasarev, A. Mélikhov, D. Pospélov, E. Yakoubaïtis et autres.

S. Baranov

Janvier 1973

EXTRAIT DE L'AVANT-PROPOS À LA DEUXIÈME ÉDITION RUSSE

Cinq ans se sont écoulés depuis la parution de la première édition de la « Synthèse des automates microprogrammés ». Bien qu'on observe ces dernières années une certaine stabilité dans le domaine des modèles mathématiques des systèmes numériques, l'auteur n'en a pas moins tenu à reviser la première édition et à la compléter de nouveaux chapitres (chapitres 1, 3, 4, 10). Le chapitre 3 est consacré au problème de la recherche du recouvrement minimal fermé, auquel se ramène non seulement le problème de minimisation d'automates partiels, mais aussi bon nombre de problèmes importants dépassant le cadre de la théorie des automates. En ajoutant à cette nouvelle édition des paragraphes traitant de la décomposition des automates (chapitre 4) l'auteur a voulu attirer l'attention sur ce domaine prometteur et stimuler les recherches dans cette direction. Les matières exposées dans le chapitre 10 sont le fruit des efforts déployés ces dernières années par de nombreux chercheurs dans le domaine de la synthèse d'automates de commande avec des matrices logiques programmables. Les éléments de théorie des ensembles (chapitre 1) sont inclus dans cette deuxième édition afin de conférer à l'ouvrage un caractère autonome du point de vue mathématique.

S. Baranov

Février 1978

AVANT-PROPOS À L'ÉDITION FRANÇAISE

En vue de la traduction en français de la « Synthèse des automates microprogrammés » (Léningrad, « Energuia », 1979), l'auteur a apporté au texte quelques corrections et modifications, a entièrement remanié le chapitre 10 et a ajouté le chapitre 11, consacrés l'un et l'autre à la réalisation matricielle et à la synthèse des automates microprogrammés avec des matrices logiques programmables.

L'inclusion de ces matières dans l'ouvrage s'imposait, vu les grands succès réalisés les 10 dernières années dans le domaine des circuits intégrés, succès qui ont abouti à l'intégration à grande échelle (IGE) — à la conception de circuits microminiaturisés dont un cristal renferme des dizaines de milliers d'éléments. Or, réaliser des dispositifs concrets sur la base d'un cristal comportant un nombre colossal d'éléments est un problème extrêmement compliqué.

Une première approche de la solution de ce problème consistait à réaliser des circuits « sur commande » (on aurait pu dire « sur mesure »), destinés à équiper tel ou tel matériel concret. La fabrication des circuits microminiaturisés sur commande étant un processus long et onéreux, faisant appel à des systèmes complexes homme — machine de synthèse automatique, on conçoit que ces techniques ne sont acceptables qu'à condition de fabriquer ce matériel en série.

Des assemblages de microprocesseurs qui sont des blocs de circuits microminiaturisés permettant de réaliser des fonctions complexes des systèmes numériques sont venus pallier les insuffisances et les défaillances des circuits IGE faits sur commande. Avec ces « briques » on peut bâtir, par des techniques beaucoup plus simples, des microcalculateurs qui ont connu ces dernières années des progrès spectaculaires et pénétré dans des systèmes de commande diversement spécialisés.

Le microprocesseur est un dispositif universel capable de réaliser diverses fonctions logiques. Cependant, la réalisation, par les moyens du logiciel, des algorithmes de commande complexes prend beaucoup

de temps, de sorte que le microprocesseur n'assure pas toujours la rapidité requise. C'est ce qui explique l'intérêt croissant pour les matrices logiques programmables (MLP). Celles-ci représentent des circuits microminiaturisés alliant la régularité de structure d'une mémoire à transistors à l'universalité du microprocesseur. Les MLP ont sur les microprocesseurs l'avantage d'assurer une réalisation plus rapide d'algorithmes de commande complexes, surtout lorsque l'information à retenir provisoirement n'est pas trop volumineuse.

L'adjectif « programmable » dans la dénomination matrices logiques programmables insiste sur la possibilité de les programmer, à la façon des mémoires à transistors. Cette méthode de structuration des MLP est proche de la méthode intuitive de synthèse des circuits logiques, lorsqu'on n'a pas recours à des modèles mathématiques formels. D'autre part, les recherches des spécialistes (dont l'auteur du présent ouvrage) ont prouvé que les méthodes de la théorie des automates non seulement permettent de simplifier et d'accélérer les travaux de conception, mais aussi de résoudre parallèlement des problèmes d'optimisation auxquels on est souvent amené au cours des travaux.

Conseils au lecteur. Les lecteurs ne recherchent évidemment pas tous le même but en abordant la lecture de notre ouvrage. Le matériel est présenté de façon à permettre au lecteur désireux d'entrer au plus vite dans le vif du sujet qui l'intéresse d'omettre en première lecture certains paragraphes et même des chapitres entiers.

Ainsi, par exemple, si l'on veut se familiariser avec la théorie des automates dite « classique », il suffit de lire les chapitres 2 à 5 ; la théorie des schémas opératoriels d'algorithmes est exposée dans les chapitres 6 et 7. Le lecteur désireux d'apprendre en premier lieu les techniques de la synthèse des automates microprogrammés complexes peut se borner à la lecture des chapitres 2, 5, 6, 8, suivant le procédé de réalisation du circuit logique qui l'intéresse. On peut de surcroît omettre, en première lecture de ces quelques chapitres, les §§ 2-3, 2-4, 5-5, 5-6, 6-4, 6-5, 6-6 et 8-3. Si l'on est intéressé uniquement par la synthèse des automates avec des MLP, les §§ 2-1, 2-2, 5-1, 5-2, 5-4, 6-1, 6-2, 8-1, 8-2, 10-1, 10-2, 11-1, 11-2 suffisent. Dans tous les cas on suppose que le lecteur connaît les éléments de théorie des ensembles, ne serait-ce que dans le volume très restreint du premier chapitre.

Conseils au professeur. L'ouvrage est issu du cours oral que l'auteur a professé pendant une dizaine d'années à l'Institut de la mécanique de précision et de l'optique de Léningrad, ainsi que dans le cadre de divers programmes de recyclage de professeurs et d'ingénieurs se spécialisant dans le domaine de l'automatique et du calcul numérique. Ce cours étalé sur deux semestres contenait tout le

matériel du livre, sauf quelques sections des chapitres 3, 4 et 7. Tout professeur sait par expérience que donner un cours concis est une gageure. Le cours le plus concis (de 16 heures) sur la synthèse des automates avec des MLP que l'auteur ait lu comprenait le matériel des §§ 2-1, 2-2, 5-1, 5-4, 6-1, 6-2, 8-1, 9-1, 10-1, 11-1, 11-2.

L'auteur exprime sa profonde reconnaissance à ses amis, collègues, élèves pour leurs remarques et suggestions. Il tient à remercier cordialement Mme L. Jouravina, candidat ès sciences techniques, pour la rédaction des programmes et l'étude de la plupart des algorithmes exposés dans l'ouvrage ainsi que pour la préparation du manuscrit à la publication.

S. Baranov

Octobre 1982

CHAPITRE PREMIER

ÉLÉMENTS DE THÉORIE DES ENSEMBLES

1-1. L'ensemble

En mathématiques modernes, la notion d'*ensemble* est une notion première, qui n'est pas introduite à partir d'autres notions. Par ensemble on entend un groupement (on dit aussi famille, collection, classe) d'objets, dits *éléments* (*membres*, ou *points*) de cet ensemble. Un ensemble est défini par ses éléments. On peut parler de l'ensemble des pages d'un livre, de l'ensemble des lettres d'une page, de l'ensemble des lecteurs du présent ouvrage (on voit d'emblée qu'un ensemble peut ne pas avoir d'éléments), de l'ensemble des nombres réels, etc. Si l'on veut donner un ensemble, l'essentiel est de donner des règles qui permettent de dire si tel objet est bien (ou non) élément de l'ensemble dont il s'agit.

Pour écrire que a est élément d'un ensemble A (on dit aussi « a appartient à A ») on utilise la notation : $a \in A$.

Le contraire, c'est-à-dire « a n'appartient pas à A », est noté $a \notin A$.

Un ensemble est dit *fini* s'il contient un nombre fini d'éléments et *infini* dans le cas contraire.

Deux ensembles A et B sont *égaux* (ou *identiques*) ($A = B$) si et seulement s'ils sont formés des mêmes éléments. Pour s'assurer de l'égalité de deux ensembles il suffit de vérifier que $a \in A$ entraîne $a \in B$ et, inversement, $b \in B$ entraîne $b \in A$.

Le plus souvent, pour représenter un ensemble, on utilise l'un des procédés suivants :

1. On énumère tous ses éléments qu'on écrit dans une accolade séparés par une virgule. Par exemple,

$$A = \{a_1, a_2, a_3\}.$$

On peut permuter les éléments d'un ensemble sans que celui-ci change. Ainsi, les ensembles $\{1, 3, 5\}$ et $\{1, 5, 3\}$ sont identiques. Il est clair que par ce procédé on ne peut donner que des ensembles finis et dont la liste des éléments n'est pas trop longue. En principe, on aurait pu donner de la sorte l'ensemble des nombres entiers de 1 à 10^9, n'était la longueur de la liste de ses éléments.

2. On donne le symbole de l'élément général, avec l'énoncé de la propriété commune à tous les éléments de l'ensemble que l'on veut définir :

$$A = \{a \mid a \text{ possède la propriété } Q\}.$$

Le trait vertical placé après a se lit « tels que », i.e. A est l'ensemble des éléments tels qu'ils possèdent la propriété Q. Par exemple, si $N = 1, 2, 3, \ldots$, alors $A = \{a \mid a = 2n,\ n \in N\}$ est l'ensemble des nombres pairs. Parfois, les ensembles infinis sont définis par leurs quelques premiers éléments ; la propriété commune à tous les éléments est alors énoncée implicitement. Ainsi, on peut donner l'ensemble des nombres pairs en écrivant $A = \{2, 4, 6, 8, \ldots\}$.

On dit qu'un ensemble A est un *sous-ensemble* d'un ensemble B (ou « A est contenu dans B », ou encore « A est une partie de B », ou « B contient A ») si et seulement si tout élément de A appartient à B. Notations :

$$A \subseteq B \text{ ou } B \supseteq A.$$

Un ensemble ne possédant aucun élément (ce qu'on exprime encore en disant que le nombre de ses éléments est nul) est appelé ensemble *vide* et noté $\varnothing$. La notion d'ensemble vide est nécessaire non seulement pour simplifier diverses définitions, mais aussi pour n'avoir pas à mentionner chaque fois, au cours des opérations sur les ensembles (voir plus bas), si tel ou tel ensemble est vide ou non. D'ailleurs, on peut ne pas le savoir d'avance. Par exemple, sans avoir regardé la page 173, on ne peut pas dire si l'ensemble des figures y est vide ou non. Pourtant, la notion d'ensemble des figures des pages 173, 174 et 175 a un sens. Autre exemple : jusqu'à présent (du moins jusqu'au moment où j'écris ces lignes et non quand mon livre sortira des presses) on ne sait pas si l'ensemble des racines entières de l'équation $x^n + y^n = z^n$ pour $n > 2$ (problème de Fermat) est vide ou non.

Par définition d'un sous-ensemble, l'ensemble vide est évidemment un sous-ensemble de n'importe quel ensemble, vu que tous ses éléments (inexistants par définition) sont des éléments de tout ensemble. Tout ensemble étant en même temps un de ses sous-ensembles ($A \subseteq A$), il en est de même de l'ensemble vide : $\varnothing \subseteq \varnothing$.

Le nombre de tous les sous-ensembles possibles d'un ensemble fini quelconque composé de n éléments est 2^n. Ainsi, l'ensemble $A = \{1, 2, 3\}$ a huit sous-ensembles : $\varnothing$, $\{1\}$, $\{2\}$, $\{3\}$, $\{1, 2\}$, $\{1, 3\}$, $\{2, 3\}$, $\{1, 2, 3\}$. L'ensemble vide $\varnothing$ et l'ensemble A lui-même sont les seuls *sous-ensembles impropres* de A, tous les autres sous-ensembles énumérés en sont les *sous-ensembles propres*.

Il faut faire la distinction entre l'élément a et l'ensemble à un élément $\{a\}$. Dans cet ordre d'idées, l'ensemble $\{\varnothing\}$ n'est pas vide ($\varnothing \neq \{\varnothing\}$), mais est un ensemble à un élément : son unique élément est l'ensemble vide. Quelques exemples :

1. $A = \{1, 2, 3, 4\}$; $1 \in A$; $3 \in A$; $\{1, 3\} \notin A$; $\{1, 3\} \subseteq A$.
2. $B = \{1, 2, \{3\}\}$; $1 \in B$; $3 \notin B$; $\{1, 2\} \subseteq B$; $\{2\} \subseteq B$; $\{3\} \nsubseteq B$; $\{3\} \in B$.
3. $C = \{1, 2, \{1, 2\}\}$; $\{1, 2\} \in C$; $\{1, 2\} \subseteq C$.
4. $D = \{\varnothing, \{\varnothing\}\}$; $\varnothing \in D$; $\{\varnothing\} \in D$; $\{\varnothing\} \subseteq D$ (en raison de $\varnothing \in D$) ; $\varnothing \subseteq D$ (l'ensemble vide étant sous-ensemble de tout ensemble).

1-2. Opérations sur les ensembles

Nous aurons besoin par la suite de la notion d'ensemble universel. Dans une théorie mathématique (et non seulement en mathématiques) il est souvent utile d'introduire en considération un ensemble U aussi vaste que tous les ensembles possibles en seront des sous-ensembles ; on l'appelle *ensemble universel*, ou *univers*, ou *référentiel*.

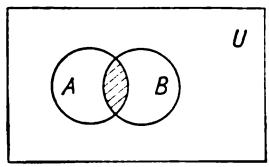


Fig. 1-1. Intersection de deux ensembles

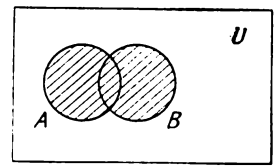


Fig. 1-2. Réunion de deux ensembles

A noter que la notion d'ensemble universel n'est pas une notion absolue : son contenu change suivant le domaine concret. Ainsi, en arithmétique, l'ensemble universel est l'ensemble R des nombres rationnels. Encore un exemple, choisi dans un domaine autre que mathématique : s'il s'agit d'appliquer telles ou telles mesures dans le cadre d'une Ecole supérieure, il est commode de prendre comme ensemble universel l'ensemble des élèves de cette Ecole et non celui de toute la ville ou de tout le pays.

L'*intersection* de deux ensembles A et B, notée $A \cap B$, se définit comme suit :

$$A \cap B = \{a \mid a \in A \text{ et } a \in B\}.$$

On représente cette opération à l'aide d'un diagramme de Venn (fig. 1-1). Le référentiel U et les ensembles A et B sont les ensembles

des points des figures planes limitées par des courbes fermées correspondantes. La zone hachurée est $A \cap B$.

La *réunion* de deux ensembles, notée $A \cup B$, se définit comme suit (fig. 1-2) :

$$A \cup B = \{a \mid a \in A \text{ ou } a \in B\}.$$

Le *complémentaire* de A dans U, noté $\overline{A}$, est une opération unaire (fig. 1-3) :

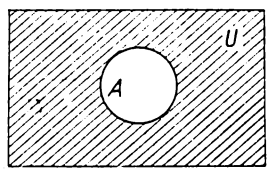
$$\overline{A} = \{a \mid a \in U \text{ et } a \notin A\}.$$

La *différence* de deux ensembles A et B, notée $A \setminus B$, se définit comme suit (fig. 1-4) :

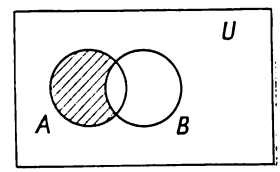
$$A \setminus B = \{a \mid a \in A \text{ et } a \notin B\}.$$

Il est clair que $\overline{A} = U \setminus A$.

On notera que toutes ces opérations sont fermées dans U : en effet, aussi bien les opérandes que les résultats des opérations appartiennent à U.



Fir. 1-3. Complémentaire de l'ensemble A



Fir. 1-4. Différence de deux ensembles

Les opérations d'intersection et de réunion étant associatives :

$$(A \cap B) \cap C = A \cap (B \cap C);$$
$$(A \cup B) \cup C = A \cup (B \cup C),$$

on définit de manière analogue l'intersection et la réunion d'un nombre quelconque d'ensembles :

$$A_1 \cap A_2 \cap \ldots \cap A_n = \bigcap_{i=1}^{n} A_i;$$

$$A_1 \cup A_2 \cup \ldots \cup A_n = \bigcup_{i=1}^{n} A_i.$$

Pour définir le produit de deux ensembles nous aurons besoin de la notion de *cortège* (vecteur), encore une notion première en mathématiques modernes. A cette notion est associée celle de *compo-*

sante (coordonnée). On appelle *couple* un cortège de deux composantes, *triplet* un cortège de trois composantes, *n-uplet* un cortège de n composantes; on définit un cortège à n composantes en énumérant, entre parenthèses, tous ses éléments. Exemple: $\alpha = (a_1, a_2, \ldots, a_n)$.

A la différence d'un ensemble dont tous les éléments sont distincts, un cortège peut comporter plusieurs occurrences d'une composante. Exemple: (1, 2, 1, 3, 2) est un cortège à cinq composantes. Le nombre de composantes d'un cortège est sa *longueur*.

Deux cortèges sont égaux lorsqu'ils sont de même longueur et ont égales composantes à des places de même numéro. Ainsi, étant donné $(a_1, a_2, \ldots, a_n) = (a_1', a_2', \ldots, a_n')$, on a $a_i = a_i'$ pour tous les $i = 1, \ldots, n$. Soit, par exemple, des cortèges $\alpha = (1, 2, 3, 2, 4)$; $\beta = (1, 2, 3, 2)$; $\gamma = (1, 2, 3, 2, 4)$; $\delta = (1, 2, 3, 1, 4)$; on voit que $\alpha = \gamma$, mais $\alpha \neq \beta$ et $\alpha \neq \delta$.

Soient A et B deux ensembles. On appelle *produit* de ces deux ensembles, et l'on note $C = A \times B$, l'ensemble

$$C = \{(a, b) \mid a \in A, \ b \in B\}$$

formé des couples (a, b). L'ensemble C comporte mn éléments, m et n étant respectivement les nombres des éléments de A et B.

L'ensemble

$$C = \{(a_1, a_2) \mid a_1 \in A \text{ et } a_2 \in A\}$$

formé des couples ordonnés des éléments de A sera appelé le *carré* de l'ensemble A et noté A^2.

Si $A_1, \ldots, A_n$ sont des ensembles, on entend par leur produit l'ensemble

$$B = A_1 \times A_2 \times \ldots \times A_n = \mathop{\times}_{i=1}^{n} A_i = \{(a_1, \ldots, a_n) \mid a_1 \in A_1, \ldots, a_n \in A_n\}$$

formé des cortèges de longueur n tels que $a_i \in A_i$. Soit $A = \{1, 2\}$, $B = \{a, b, c\}$. On a $A \times B = \{(1, a), (1, b), (1, c), (2, a), (2, b), (2, c)\}$. Il est clair que $A \times B \neq B \times A$; $A \times (B \times C) \neq (A \times B) \times C \neq A \times B \times C$; $A^{r_1} \times A^{r_2} \neq A^{r_1+r_2}$.

1-3. Correspondance. Fonction. Relation

On appelle *correspondance binaire* entre un ensemble A et un ensemble B un sous-ensemble P tel que $P \subseteq A \times B$. L'ensemble A est dit *domaine de départ* et l'ensemble B, *domaine d'arrivée* de la correspondance P.

Dire qu'il y a correspondance entre A et B, c'est dire qu'à des éléments de A (éventuellement à tous les éléments de A) correspon-

dent des éléments de B. Soit, par exemple, $A = \{1, 2, 3, 4, 5\}$; $B = \{a, b, c, d, e, f\}$; définissons une correspondance P comme suit: $P = \{(1, a), (3, a), (3, b), (3, c), (4, c), (5, b), (5, e)\}$. Les correspondances sont susceptibles de plusieurs représentations, nous en allons utiliser deux : par tableaux (fig. 1-5) et par flèches (fig. 1-6).

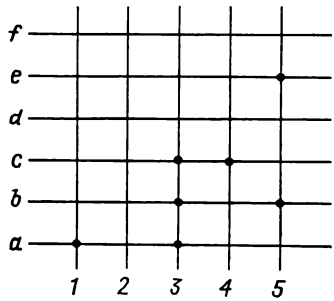


Fig. 1-5. Graphe de la correspondance P

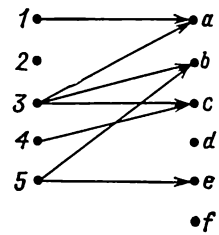


Fig. 1-6. Représentation par flèches de la correspondance P

Soit A un ensemble. On appelle *relation binaire* sur A un sous-ensemble R tel que $R \subseteq A^2$, i.e. un sous-ensemble du carré de A. La relation binaire est donc un cas particulier de la correspondance où l'ensemble de départ est égal à l'ensemble d'arrivée. Si D est

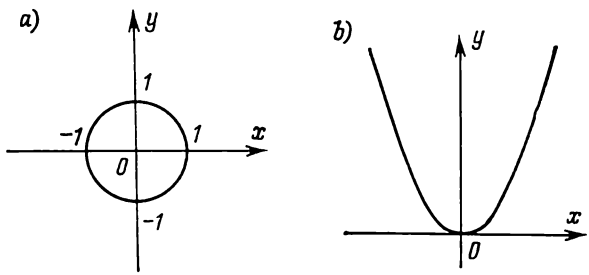


Fig. 1-7. Courbes des relations R_1 (a) et R_2 (b)

l'ensemble des points d'une droite, D^2 est un plan. Une relation quelconque sur D peut être considérée comme un ensemble de points marqués sur ce plan. Deux relations $R_1 = \{(x, y) \mid x^2 + y^2 = 1\}$ et $R_2 = \{(x, y) \mid y = x^2\}$ sont représentées graphiquement sur la figure 1-7, a et b respectivement.

Il y a deux correspondances triviales entre deux ensembles A et B :

— l'ensemble $A \times B$ (partie pleine de $A \times B$);

— le sous-ensemble vide $\varnothing \subseteq A \times B$ (*correspondance vide*).

Deux correspondances P, Q entre A et B peuvent vérifier une relation d'inclusion $P \subseteq Q$. Les correspondances étant des ensembles, on définit la notion d'égalité de correspondances: deux correspondances P et Q sont égales si et seulement si elles sont constituées par les mêmes éléments. On peut effectuer sur les correspondances, considérées comme sous-ensembles de $A \times B$, les opérations $\cap$, $\cup$, $\setminus$, la complémentation. En tant que sous-ensembles de $A \times B$ ou de A^2, et non des ensembles quelconques, les correspondances et relations possèdent quelques propriétés spécifiques que nous allons indiquer. Tout ce qui sera dit des correspondances se rapporte aux relations, qui en sont des cas particuliers.

La *première* (*seconde*) *projection* d'un couple $\tilde{a} = (a, b)$ est égale à sa première (seconde) coordonnée:

$$\mathrm{pr}_1(\tilde{a}) = a; \ \mathrm{pr}_2(\tilde{a}) = b.$$

Si $P \subseteq A \times B$ est une correspondance, ses projections $\mathrm{pr}_1(P)$ et $\mathrm{pr}_2(P)$ sont définies comme suit:

$$\mathrm{pr}_1(P) = \{a \mid a = \mathrm{pr}_1(\tilde{a}) \text{ pour } \tilde{a} \in P\};$$

$$\mathrm{pr}_2(P) = \{b \mid b = \mathrm{pr}_2(\tilde{a}) \text{ pour } \tilde{a} \in P\}.$$

Pour la correspondance $P = \{(1, a), (3, a), (3, b), (3, c), (4, c), (5, b), (5, e)\}$ (fig. 1-5 ou 1-6) les projections première et seconde sont $\mathrm{pr}_1(P) = \{1, 3, 4, 5\}$; $\mathrm{pr}_2(P) = \{a, b, c, e\}$.

Etant donnée une correspondance $P \subseteq A \times B$, on appelle $\mathrm{pr}_1(P)$ son *domaine* (ensemble) *de définition*, et $\mathrm{pr}_2(P)$ son *domaine* (ensemble) *des valeurs*. Si $A = \mathrm{pr}_1(P)$ (i.e. le domaine d'arrivée est égal au domaine de définition), on dit que la correspondance P est *partout* (complètement) *définie*. Si $B = \mathrm{pr}_2(P)$, on dit que l'ensemble B est le domaine des valeurs de P ou que la correspondance P est *surjective*.

Soit $P \subseteq A \times B$ une correspondance; $X = \mathrm{pr}_1(P)$, $Y = \mathrm{pr}_2(P)$. On dit que P est définie dans A et prend ses valeurs dans B; X est l'ensemble de définition de P et Y, son ensemble des valeurs.

On appelle *image* d'un élément A par la correspondance $P \subseteq \subseteq A \times B$, ou la *coupe gauche* de la correspondance P par l'élément a, l'ensemble

$$S_l(a, P) = \{b \mid b \in B \text{ et } (a, b) \in P\}.$$

On appelle *image réciproque* d'un élément $b \in B$ par la correspondance $P \subseteq A \times B$, ou la *coupe droite* de la correspondance P par l'élément b, l'ensemble

$$S_r(P, b) = \{a \mid a \in A, \text{ et } (a, b) \in P\}.$$

Si $A' \subseteq A$, on appelle image de l'ensemble A' par la correspondance $P \subseteq A \times B$ (coupe gauche de la correspondance P par l'ensemble A') l'ensemble

$$S_l(A', P) = \bigcup_{a \in A'} S_l(a, P).$$

Si $B' \subseteq B$, on appelle image réciproque de l'ensemble B' par la correspondance $P \subseteq A \times B$ (coupe droite de P par l'ensemble B') l'ensemble

$$S_r(P, B') = \bigcup_{b \in B'} S_r(P, b).$$

Pour la correspondance considérée (fig. 1-5, 1-6), $S_l(3, P) = \{a, b, c\}$; $S_l(1, P) = \{a\}$; $S_l(2, P) = \varnothing$; $S_r(P, a) = \{1, 3\}$; $S_r(P, d) = S_r(P, f) \neq \varnothing$. Si $A' = \{1, 2, 4\} \subseteq A$; $B' = \{b, c, d\} \subseteq B$, alors $S_l(A', P) = \{a, c\}$; $S_r(P, B') = \{3, 4, 5\}$.

Soit $\tilde{a} = (a, b) \in A \times B$. Le couple $\tilde{a}^{-1} = (b, a) \in B \times A$ est dit *symétrique* du couple $\tilde{a}$; l'opération par laquelle on fait correspondre à des couples $\tilde{a} \in A \times B$ les couples symétriques porte le nom de *symétrisation*.

Si $P \subseteq A \times B$, la symétrique de P, notée P^{-1}, n'est autre que le sous-ensemble de $B \times A$ formé des couples symétriques des couples de P:

$$P^{-1} = \{(b, a) \mid a \in A,\ b \in B,\ (a, b) \in P\}.$$

Dans notre exemple, $P^{-1} = \{(a, 1), (a, 3), (b, 3), (b, 5), (c, 3), (c, 4), (e, 5)\}$. On constate que par symétrisation, la première projection de P devient la seconde projection de P^{-1}, les coupes gauches deviennent droites, les coupes droites, gauches, et ainsi de suite.

Soient trois ensembles A, B, C et deux correspondances $P \subseteq A \times B$ et $Q \subseteq B \times C$. La correspondance $R \subseteq A \times C$, dite *composée* de P et Q, se définit comme suit:

$$R = P \circ Q = \{(a, c) \mid \text{il existe un } b \in B \text{ tel que } (a, b) \in P \text{ et } (b, c) \in Q\}.$$

Exemple. $A = \{1, 2, 3, 4\}$; $B = \{a, b, c\}$; $C = \{u, v, w, x, y\}$ et $P = \{(1, b), (1, c), (2, a), (4, a), (4, c)\}$; $Q = \{(a, u), (a, x), (b, w), (c, v), (c, y)\}$; $R = P \circ Q = \{(1, w), (1, v), (1, y), (2, u), (2, x), (4, u), (4, x), (4, v), (4, y)\}$. Il est évident que la composée $P \circ Q$ n'est pas vide que si $\mathrm{pr}_2 P \cap \mathrm{pr}_1 Q \neq \varnothing$. Dans le cas contraire elle est indéterminée.

On montre facilement que la composition des correspondances est associative, i.e. $(P \circ Q) \circ T = P \circ (Q \circ T)$, P, Q, T étant des correspondances.

Soient A un ensemble et $R \subseteq A^2$ une relation binaire. Construisons un graphe ayant pour sommets les éléments de A; deux sommets

a_i et a_j seront réunis par une flèche dirigée de a_i vers a_j si $(a_i, a_j) \in R$. La figure 1-8 montre le graphe de la relation $P = \{(1, 1), (1, 2), (1, 3), (2, 3), (3, 2), (3, 3)\}$ définie sur l'ensemble $A = \{1, 2, 3\}$. Il est clair qu'à chaque relation correspond un graphe unique défini comme ci-dessus, par conséquent, la théorie des graphes orientés et la théorie des relations binaires sont au fond une même théorie.

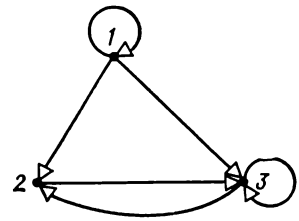


Fig. 1-8. Exemple de graphe d'une relation binaire

Une correspondance $\varphi \subseteq A \times B$ est dite *fonctionnelle*, ou *fonction*, lorsque la coupe gauche de φ par tout élément comprend au plus un élément. On voit que la correspondance de la figure 1-6 n'est pas une fonction, puisque les éléments 3 et 5 de son ensemble de définition ont plus d'une image. Pour la même raison, la correspondance montrée sur la figure 1-7, *a* n'est pas une fonction non plus : chaque élément de son domaine de définition, sauf $x = -1$ et $x = 1$, admet deux images. Quant à la correspondance de la fig. 1-7, *b*, elle est bien une fonction.

Une fonction est dite *définie partout*, ou *application*, lorsque son ensemble de définition se confond avec son ensemble de départ : $A = \mathrm{pr}_1\, \varphi$. Dans le présent ouvrage, les fonctions seront désignées comme suit :

$$\varphi : A \to B\,; \quad A \xrightarrow{\varphi} B\,; \quad \varphi\,(x)\,; \quad y = \varphi\,(x).$$

On dit que φ est une *surjection* si et seulement si son ensemble d'arrivée est égal à son ensemble des valeurs : $B = \mathrm{pr}_2\, \varphi$. On dit que φ est une *injection* si deux éléments distincts de A ont des images distinctes : si $x_1 \neq x_2$, alors $f\,(x_1) \neq f\,(x_2)$. On dit que φ est une *bijection*, ou une *application bijective*, si φ est à la fois injective et surjective. On l'appelle souvent également *application biunivoque*.

1-4. Relations d'équivalence et d'ordre

Soient A un ensemble et R une relation binaire dans A. Au lieu d'écrire $(a, b) \in R$ nous ferons parfois l'usage de la notation aRb. Citons quelques propriétés des relations binaires :

1. R est *réflexive* si aRa pour tous les $a \in A$;
2. R est *antiréflexive* *) si $(a, a) \notin R$ pour tous les $a \in A$;
3. R est *symétrique* si aRb entraîne bRa ;
4. R est *antisymétrique* si aRb et bRa entraînent $a = b$;

*) Il ne faut pas confondre les notions de relation non réflexive et de relation antiréflexive. Dans le premier cas, la relation aRa peut être vraie pour certains $a \in A$ (mais pas pour tous).

5. R est *transitive* si aRb et bRc entraînent aRc.

Si une relation binaire R est à la fois réflexive, symétrique et transitive, on dit que R est une *relation d'équivalence*. Exemples de relations d'équivalence :

1. La relation d'égalité dans A, dite *diagonale* de A.
2. La relation binaire $R = A^2$ (partie pleine).
3. La relation « faire la même année » dans l'ensemble des étudiants d'une faculté.
4. La relation « donner le même reste de la division par 3 » (congruence mod 3).
5. Soit une application $f : X \to Y$. Définissons dans l'ensemble X une relation R telle que $x_1 R x_2$ ait lieu si et seulement si $f(x_1) = f(x_2)$. On montre facilement que R est une relation d'équivalence.

Chaque relation d'équivalence donnée sur un ensemble A définit une *partition* de A en *classes d'équivalence*, sous-ensembles non vides de A, disjoints deux à deux et dont la réunion est l'ensemble A. Autrement dit, $\{B_1, \ldots, B_n\}$ est une partition de A ($B_i \subseteq A$), si et seulement si on a :

1. $A = \bigcup_{i=1}^{n} B_i$;
2. $B_i \cap B_j = \varnothing$ pour $i \neq j$;
3. $B_i = \varnothing$.

La notion de partition permet de définir celle de relation d'équivalence : une relation R définie sur un ensemble A est une relation d'équivalence si existe une partition $\{B_1, \ldots, B_n\}$ de l'ensemble A telle que aRb est vraie si et seulement si a et b sont éléments d'un même sous-ensemble de cette partition. On dit dans ce cas que R est la relation d'équivalence pour la partition considérée, tandis que $B_1, \ldots, B_n$ sont les classes d'équivalence suivant R. Nous reviendrons aux partitions pour les traiter en détail au chapitre 4.

Une relation binaire R définie sur un ensemble A est appelée *relation de préordre*, ou *relation d'ordre partiel*, si elle est réflexive, transitive et antisymétrique. Notation : $\leqslant$. Exemples d'ordre partiel :

1. La relation $\leqslant$ (au plus égal) sur l'ensemble des nombres entiers.
2. La relation $\subseteq$ (inclusion) sur l'ensemble des parties d'un ensemble. Exemple : soit $A = \{1, 2, 3\}$, l'ensemble de ses parties comporte 8 éléments : $\{\{\varnothing\}, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$. On a les inclusions : $\{1\} \subseteq \{1, 2\}$; $\{1\} \subseteq \{1, 3\}$; $\{1, 2\} \subseteq \{1, 2, 3\}$; $\varnothing \subseteq$ (toute partie de A) ; $\{1, 3\} \subseteq \{1, 3\}$, mais $\{1\} \nsubseteq \{2\}$; $\{1, 2\} \nsubseteq \{1, 3\}$, etc.
3. La relation « être descendant » sur l'ensemble des hommes existant ou ayant existé sur Terre.

4. L'hiérarchie sur l'ensemble des postes d'une entreprise.

5. La relation « être diviseur de » sur l'ensemble des nombres naturels.

En plus d'être réflective, antisymétrique et transitive, une relation R peut vérifier la condition de *dichotomie*: pour deux éléments quelconques a, $b \in A$, on a soit aRb, soit bRa. La relation R est alors une relation *d'ordre total*. Dans ce cas, deux éléments quelconques de l'ensemble sur lequel est définie une relation d'ordre total sont ordonnés.

Une relation antiréflexive, antisymétrique et transitive est une relation *d'ordre strict* (notation: $<$). Si deux éléments quelconques de l'ensemble sur lequel on a défini une relation d'ordre strict sont ordonnés, la relation en question est une relation *d'ordre total strict*. Une telle relation vérifie la condition de *trichotomie*: pour tout couple d'éléments a, $b \in A$ on a soit $a < b$, soit $b < a$, soit $a = b$ (a et b est même élément).

Un ensemble A sur lequel on a défini une relation de préordre est dit *ensemble partiellement ordonné*, ce que l'on note $(A, \leqslant)$. On définit de façon analogue des ensembles totalement ordonnés, des ensembles strictement ordonnés (notation: $(A, <)$) et des ensembles totalement strictement ordonnés. Donnons encore quelques exemples de relations d'ordre.

6. La relation d'inclusion stricte ($\subset$) sur l'ensemble des parties d'un ensemble est une relation d'ordre stricte. Pour l'ensemble de l'exemple 2, on a $\{1\} \subset \{1, 2\}$; $\{1, 2\} \subset \{1, 2, 3\}$; $\varnothing \subset$ (toute partie de A), mais $\{1, 3\} \not\subset \{1, 2\}$; $\{1\} \not\subset \{2\}$; $\{1, 2\} \not\subset \{1, 3\}$, etc.

7. La relation $<$ (strictement inférieur à) sur l'ensemble des entiers est une relation d'ordre total strict.

8. Soit $(A, \leqslant)$ un ensemble totalement ordonné. Définissons sur l'ensemble A^n de vecteurs à n composantes une relation d'ordre lexicographique L de la façon suivante. Soient $c_p = (a_1, \ldots, a_n)$; $c_r = (b_1, \ldots, b_n)$; c_p, $c_r \in A^n$; a_i, $b_i \in A$, $i = 1, \ldots, n$. Alors $c_p L c_r$ si pour le plus petit indice i tel que $a_i \neq b_i$ on a $a_i < b_i$. Il est facile de voir que L est une relation d'ordre total sur l'ensemble A^n.

Par exemple, si $A = \{0, 1, \ldots, 9\}$, alors l'ensemble A^5 des entiers positifs à cinq chiffres est muni d'une relation d'ordre total $\leqslant$ (un nombre formé de moins de cinq chiffres est supposé complété à gauche par des zéros jusqu'à cinq chiffres):

$$11249 \leqslant 11324; \quad 00937 \leqslant 01012 \text{ et ainsi de suite.}$$

9. Soit $(A, \leqslant)$ un ensemble totalement ordonné. L'ensemble $A \cup A^2 \cup A^3 \cup \ldots \cup A^n$ de vecteurs de différente longueur peut être muni d'un *ordre lexicographique*. Si A est l'ensemble des lettres

de l'alphabet français rangées dans l'ordre alphabétique, les mots d'un dictionnaire sont disposés dans l'ordre lexicographique :

poil $\leqslant$ point ; point $\leqslant$ pointe ; pointe $\leqslant$ pointure, etc.

10. Considérons l'exemple d'une relation d'ordre tiré d'une nouvelle de M. Twain. Dans un ensemble A dont les éléments sont un père (P), son fils adulte (F), une jeune veuve qui est la mère (M) d'une grande fille (D), on donne une relation d'ordre R telle que aRb est vraie si et seulement si b est enfant de a. Cette relation est représentée par flèches à la figure 1-9, *a*.

Un jour F fit connaissance avec la jeune veuve M et l'épousa peu après. Le graphe de la relation R a pris l'aspect de la figure 1-9, *b*,

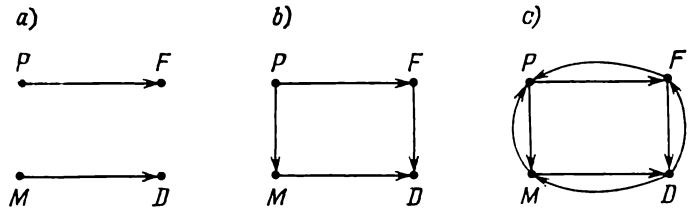


Fig. 1-9. Graphes de certaines relations illustrant le récit de M. Twain

car P devint en même temps père de M qui avait épousé son fils, et D devint fille de F, étant fille de son épouse.

Quelque temps après, P et D tombaient amoureux l'un de l'autre et se mariaient. Cette nouvelle union eut pour effet de transformer le graphe de la relation R comme montré à la figure 1-9, *c*. Mark Twain qui fait raconter cette histoire par F conclut à juste raison que F est devenu grand-père à lui-même. On le voit immédiatement à partir du graphe de la figure 1-9, *c*, puisque le chemin de F en F est de longueur deux (le chemin de grand-père à petit-fils dans une telle relation). Ce graphe montre également que chacun des quatre personnages est devenu aïeul à lui-même.

Le lecteur trouvera un grand nombre d'exemples de correspondances, fonctions et relations dans les ouvrages [55, 56, 31, 23] qui sont d'excellents ouvrages d'initiation à la théorie des ensembles.

CHAPITRE 2

AUTOMATE ABSTRAIT

2-1. Définition d'un automate abstrait

Le modèle mathématique d'un réseau séquentiel est un *automate abstrait.* C'est un cortège, ou vecteur, ou collection de six êtres mathématiques, noté

$$S = (A, Z, W, \delta, \lambda, a_1)$$

où

$A = \{a_1, \ldots, a_m, \ldots, a_M\}$ est l'ensemble des états internes (l'alphabet des états);

$Z = \{z_1, \ldots, z_f, \ldots, z_F\}$, l'ensemble des signaux d'entrée (l'alphabet d'entrée);

$W = \{w_1, \ldots, w_g, \ldots, w_G\}$, l'ensemble des signaux de sortie (l'alphabet de sortie);

$\delta: A \times Z \rightarrow A$, la *fonction des transitions* qui réalise l'application $D_\delta \subseteq A \times Z$ dans A. Autrement dit, la fonction δ fait correspondre à certains couples état interne — signal d'entrée (a_m, z_f) des états $a_s = \delta(a_m, z_f)$ de l'automate, $a_s \in A$;

$\lambda: A \times Z \rightarrow W$, la fonction des sorties qui réalise l'application $D_\lambda \subseteq A \times Z$ sur W. A certains couples état interne — signal d'entrée, la fonction λ fait correspondre des signaux de sortie de l'automate $w_g = \lambda(a_m, z_f)$;

$a_1 \in A$, l'état initial de l'automate.

Par *alphabet*, on entend ici un ensemble non vide de symboles distincts deux à deux. Les éléments de l'alphabet sont appelés lettres, et une suite ordonnée finie de lettres, séquence ou mot de l'alphabet considéré. Nous considérerons aussi le mot vide (noté e): c'est une suite de lettres vide (suite de longueur nulle).

Un automate *) (fig. 2-1) comprend une entrée et une sortie. Il fonctionne dans un temps discret qui prend des valeurs entières non négatives $t = 0, 1, 2, \ldots$ A chaque instant t du temps discret, l'automate est dans un état $a(t)$ appartenant à son ensemble des

*) A partir d'ici et jusqu'au chapitre 5, nous dirons « automate » pour « automate abstrait ».

états ; à l'instant initial $t = 0$ il se trouve toujours à l'état initial $a(0) = a_1$. Mis à l'état $a(t)$ à l'instant t, l'automate accepte à l'entrée une lettre $z(t)$ de l'alphabet d'entrée Z. Conformément à sa fonction des sorties, il produit au même instant t la lettre $w(t) = \lambda(a(t), z(t))$ de l'alphabet de sortie ; conformément à sa fonction des transitions, il passe à l'état suivant $a(t+1) = \delta(a(t), z(t))$; $a(t) \in A$, $w(t) \in W$. Un automate abstrait réalise donc une application de l'ensemble des mots de l'alphabet d'entrée Z dans l'ensemble des mots de l'alphabet de sortie W. Autrement dit, acceptant à son entrée une suite de lettres (ou séquence) de l'alphabet d'entrée : $z(0), z(1), z(2), \ldots$ (mot d'entrée), l'automate se trouvant à l'état initial a_1 produit à sa sortie une séquence de lettres de l'alphabet de sortie $w(0), w(1), w(2), \ldots$ (mot de sortie). Associant à chaque mot d'entrée le mot de sortie qui lui correspond, nous obtenons l'application φ induite par l'automate abstrait.

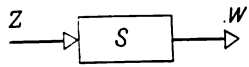


Fig. 2-1. Automate abstrait

Ainsi donc, au niveau de la théorie abstraite, le « travail » de l'automate se réduit à transformer les mots d'entrée en mots de sortie. En effet, nous ne cherchons nullement à connaître le contenu du rectangle de la figure 2-1 qui symbolise l'automate abstrait, c'est-à-dire que nous nous désintéressons de sa structure, l'assimilant à une « boîte noire » (terme admis en cybernétique pour indiquer que l'attention principale est accordée au comportement du système vis-à-vis du milieu). Dans le texte des chapitres 2 à 4, « automate » ne veut dire rien d'autre que « transformateur de mots ». Par contre, dans le chapitre 5 nous nous occuperons de près du « contenu » de la boîte : autrement dit, nous nous demanderons comment réaliser avec les éléments disponibles le comportement défini au niveau de l'automate abstrait. En refusant de considérer la structure de l'automate, on se facilite grandement la tâche : en effet, au niveau structurel, le fond du problème se perd derrière une foule de détails peu significatifs du point de vue du comportement du système dans son ensemble. On peut dire que la notion d'automate abstrait en théorie des réseaux séquentiels relève de l'approche systémique dans laquelle l'objet ou le phénomène est considéré comme un tout, on cherche à mettre en évidence ses relations et liaisons variées avec le milieu, à l'opposé de l'approche structurelle où les propriétés de l'objet sont définies par celles de ses constituants.

L'introduction de la notion d'état dans la définition d'un automate tient à ce qu'on a souvent besoin de décrire le comportement d'un système dont les sorties ne dépendent pas seulement de l'état des entrées à l'instant donné mais aussi de l'histoire des entrées, c'est-à-dire des signaux qui ont attaqué les entrées du système précédemment. C'est précisément l'état qui permet de mémoriser le passé,

éliminant le temps comme variable explicite et permettant de présenter le signal de sortie comme une fonction de l'état et de l'entrée à l'instant donné.

Dans ce livre, seront considérés principalement les automates, dits automates à mémoire, qui admettent au moins deux états. Il existe une classe d'automates où la sortie est indépendante de l'histoire, n'étant définie à tout instant que par le signal d'entrée dans ce même instant : on les appelle *réseaux combinationnels*. Au niveau abstrait, on peut les considérer comme des automates à un seul état. De toute évidence, un réseau combinationnel peut aussi être représenté comme il est montré sur la figure 2-1 et défini par un triplet $S = (Z, W, \lambda)$, où Z et W sont respectivement l'alphabet d'entrée et de sortie et $\lambda : Z \rightarrow W$ est la fonction des sorties. Un réseau combinationnel est en fait un traducteur de fonction ; nous le représenterons comme sur la figure 2-2.

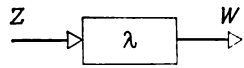


Fig. 2-2. Traducteur de fonctions

2-2. Automates de Mealy et de Moore. Les différents modes de définition d'un automate

Les deux classes d'automates le plus souvent rencontrés dans la pratique sont les *automates* (ou machines) *de Mealy* et *de Moore*, du nom des chercheurs américains G. H. Mealy et E. F. Moore qui les ont étudiés les premiers. Le fonctionnement d'un automate de Mealy se définit par les équations

$$a(t+1) = \delta(a(t), z(t)) ; \; w(t) = \lambda(a(t), z(t)), \quad t = 0, 1, 2, \ldots, \tag{2-1}$$

et le fonctionnement d'un automate de Moore, par les équations

$$a(t+1) = \delta(a(t), z(t)) ; \; w(t) = \lambda(a(t)), \; t = 0, 1, 2 \ldots \tag{2-2}$$

On dit d'un automate qu'il est *fini* si les ensembles A, Z, W le sont. Dans le texte qui suit, le mot « fini » sera d'ailleurs omis presque partout, du fait qu'on ne s'occupe que d'automates finis. Nous disons que l'automate est *complètement spécifié* (complètement défini) si l'on a $D_\delta = D_\lambda = A \times Z$. Autrement dit, pour un automate complètement spécifié, les domaines de définition des fonctions δ et λ se confondent avec l'ensemble $A \times Z$, ou ensemble des couples du type (a_m, z_f). Par contre, pour un automate *incomplètement spécifié* (partiellement spécifié ou partiel), les fonctions δ ou λ ne sont pas définies pour tous les couples $(a_m, z_f) \in A \times Z$.

Définir un automate fini S revient à définir toutes les composantes du vecteur $S = (A, Z, W, \delta, \lambda, a_1)$, c'est-à-dire l'alphabet d'entrée, l'alphabet de sortie, l'alphabet des états internes, ainsi

que les fonctions des transitions et des sorties. Parmi tous les états de l'automate, on indique celui qui correspond à l'instant $t = 0$ (l'état a_1). Le fonctionnement d'un automate peut être décrit de plusieurs façons. Le plus souvent, on a recours à un graphe ou à une table.

La définition du fonctionnement d'un automate par ses tables des transitions et des sorties est illustrée sur l'exemple d'un auto-

Tableau 2-1

δ: $A \times Z \rightarrow A$

	a_1	a_2	a_3
z_1	a_2	a_3	a_2
z_2	a_3	a_2	a_1

Tableau 2-2

λ: $A \times Z \rightarrow W$

	a_1	a_2	a_3
z_1	w_1	w_3	w_3
z_2	w_2	w_1	w_1

mate de Mealy S_1 (tabl. 2-1 et 2-2). Les lignes des tableaux correspondent aux signaux d'entrée, et les colonnes, aux états. La première colonne de gauche est réservée à l'état initial a_1. A l'intersection de la colonne a_m et de la ligne z_f, la table des transitions présente l'état $a_s = \delta(a_m, z_f)$ en lequel l'automate passe à partir de l'état a_m sous l'action du signal z_f; la table des sorties présente au même endroit le signal de sortie $w_g = \lambda(a_m, z_f)$ qui correspond à cette transition.

Tableau 2-3

δ: $A \times Z \rightarrow A$, λ: $A \rightarrow W$

	w_1	w_1	w_3	w_2	w_3
	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_5	a_5	a_3	a_3
z_2	a_4	a_2	a_2	a_1	a_1

Puisque, dans un automate de Moore, le signal de sortie ne dépend que de l'état interne, il se définit par une table unique, dite table des transitions *marquée*, dans laquelle chaque colonne est attribuée non seulement à un état a_m mais aussi au signal de sortie $w_g = \lambda(a_m)$ qui lui correspond. La définition d'un automate de Moore S_2 au moyen d'une table est illustrée par le tableau 2-3.

Dans les tables définissant un automate de Mealy partiel, dont les fonctions δ ou λ ne sont pas définies pour tous les couples $(a_m, z_f) \in A \times Z$, on met un tiret à la place de l'état et du signal de sortie non défini (voir tableaux 2-4, 2-5, d'un automate S_3). De même, pour un automate de Moore où la fonction $\lambda(a_m)$ n'est pas définie, on met un tiret au lieu du signal de sortie au-dessus de l'état a_m.

Tableau 2-4

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4
z_1	a_2	a_3	a_4	—
z_2	a_3	—	a_2	a_2

Tableau 2-5

λ: $A \times Z \to W$

	a_1	a_2	a_3	a_4
z_1	w_1	w_3	w_3	—
z_2	w_2	—	—	w_2

Quelquefois l'automate de Mealy se définit par une table unique, dite *table composite* des transitions et des sorties; à l'intersection de la colonne a_m et de la ligne z_f, une table de ce type contient l'état futur et le signal de sortie élaboré pendant la transition, soit a_s/w_g, où $a_s = \delta(a_m, z_f)$ et $w_g = \lambda(a_m, z_f)$ (voir le tableau 2-6 qui spécifie l'automate S_3).

Tableau 2-6

δ: $A \times Z \to A$, λ: $A \times Z \to W$

	a_1	a_2	a_3	a_4
z_1	a_2/w_1	a_3/w_3	a_4/w_3	—
z_2	a_3/w_2	—	$a_2/$—	a_2/w_2

Un automate peut aussi se définir à l'aide d'un *graphe orienté* dont les sommets correspondent aux états, et les arcs aux transitions. Deux sommets a_m, a_s du graphe de l'automate (l'état initial et l'état de transition) sont liés par un arc incident à a_m vers l'extérieur et à a_s vers l'intérieur si l'automate passe de l'état a_m à l'état a_s, c'est-à-dire si l'on a $a_s = \delta(a_m, z_f)$ pour un $z_f \in Z$. L'arc (a_m, a_s) du graphe est marqué par le signal d'entrée z_f, ainsi que par le signal

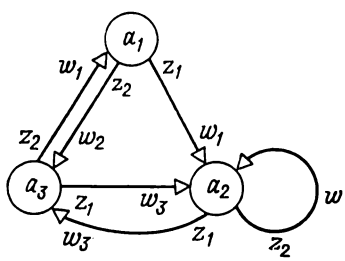


Fig. 2-3. Graphe de l'automate de Mealy S_1

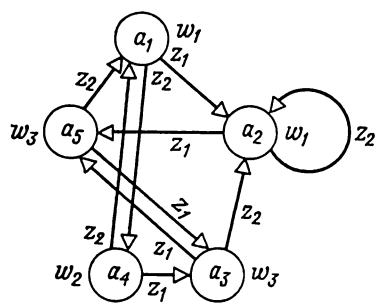


Fig. 2-4. Graphe de l'automate de Moore S_2

de sortie $w_g = \lambda(a_m, z_f)$ si ce dernier est défini, sinon on met un tiret. Si la transition de l'état a_m à l'état a_s se produit sous l'action de plusieurs signaux d'entrée, l'arc (a_m, a_s) est marqué par tous ces signaux d'entrée, ainsi que par les signaux de sortie correspondants. En définissant un automate de Moore sous forme de graphe, on écrit

le signal de sortie $w_g = \lambda(a_m)$ à l'intérieur du sommet a_m ou à côté de celui-ci. On voit sur les figures 2-3, 2-4 et 2-5 les graphes des automates S_1, S_2 et S_3 définis précédemment à l'aide des tables.

Dans le présent ouvrage, nous n'examinerons que des automates déterministes, dont toutes les transitions sont univoques, c'est-à-dire que sous l'action d'un signal d'entrée, l'automate se trouvant à un état donné ne peut passer qu'à un état bien déterminé. Tout automate défini à l'aide de sa table des transitions est forcément déterministe : il ne peut y avoir, à l'intersection d'une colonne a_m et d'une ligne z_f, qu'un état unique $a_s = \delta(a_m, z_f)$ si la transition est définie ou un tiret si la fonction δ n'est pas définie pour le couple (a_m, z_f). En parlant des transitions univoques à propos d'un automate défini sous forme de graphe, on sous-entend que deux ou plusieurs arcs marqués par un même signal d'entrée ne peuvent jamais partir du même sommet.

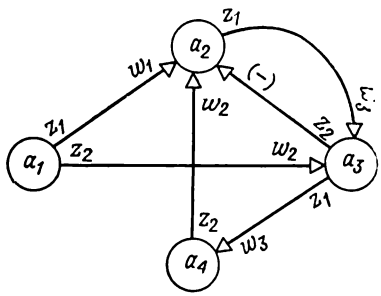


Fig. 2-5. Graphe de l'automate de Mealy partiel S_3

L'indication de l'état initial parmi tous les états de l'automate s'explique par des raisons d'utilité pratique : il est souvent commode de connaître les conditions de début de fonctionnement du réseau. Bien des problèmes se laissent résoudre par ailleurs au niveau d'un automate abstrait défini par un quintuple $S = (A, Z, W, \delta, \lambda)$. Un automate défini par un sextuple $S = (A, Z, W, \delta, \lambda, a_1)$, c'est-à-dire comprenant un état initial spécifié, s'appelle automate *initialisé*.

Généralisons les fonctions δ et λ à l'ensemble des couples état interne — mot d'entrée. Soient $\xi = z_{i_1}, \ldots, z_{i_k}$ un mot d'entrée de longueur k, E l'ensemble des mots d'entrée finis de longueur non nulle, e un mot d'entrée de longueur nulle (le mot vide). Ceci posé, on définit la *fonction d'état final* $\tilde{\delta}(a_m, \xi)$ par récurrence sur l'ensemble $A \times (E \cup \{e\})$ de la façon suivante :

$$\tilde{\delta}(a_m, e) = a_m \text{ pour tous les } a_m \in A\,;$$

$$\tilde{\delta}(a_m, \xi) = \tilde{\delta}(a_m, z_{i_1} \ldots z_{i_k}) =$$

$$= \begin{cases} \delta(\underbrace{\tilde{\delta}(a_m, z_{i_1} \ldots z_{i_{k-1}})}_{a_{i_k}}, z_{i_k}) = \delta(a_{i_k}, z_{i_k}) \text{ si la fonction } \delta(a_{i_j}, z_{i_j}) \\ \qquad \text{est définie pour tout } j = 1, \ldots, k\,;\ a_{i_1} = a_m\,; \\ \text{non définie dans le cas contraire.} \end{cases}$$

Prenons, à titre d'exemple, l'automate partiel S_3 défini par le tableau 2-6 et cherchons $\tilde{\delta}(a_2, \xi)$ pour $\xi = z_1z_1z_2z_1z_2$:

$$\tilde{\delta}(a_2, z_1z_1z_2z_1z_2) = \tilde{\delta}(\delta(a_2, z_1), z_1z_2z_1z_2) = \tilde{\delta}(a_3, z_1z_2z_1z_2) =$$
$$= \tilde{\delta}(\delta(a_3, z_1), z_2z_1z_2) = \tilde{\delta}(a_4, z_2z_1z_2) = \tilde{\delta}(\delta(a_4, z_2), z_1z_2) =$$
$$= \tilde{\delta}(a_2, z_1z_2) = \tilde{\delta}(\delta(a_2, z_1), z_2) = \tilde{\delta}(a_3, z_2) = \delta(a_3, z_2) = a_2.$$

Or, pour le même automate S_3, la fonction $\tilde{\delta}(a_2, z_1z_1z_2z_2z_2) = \tilde{\delta}(\delta(a_2, z_1), z_1z_2z_2z_2) = \tilde{\delta}(\delta(a_3, z_1), z_2z_2z_2) = \tilde{\delta}(\delta(a_4, z_2), z_2z_2) = \tilde{\delta}(\delta(a_2, z_2), z_2)$ est non définie, parce que n'est pas définie la fonction des transitions $\delta(a_2, z_2)$.

Ainsi donc, $\tilde{\delta}(a_m, \xi)$ est l'état final atteint par l'automate se trouvant en l'état a_m après une séquence d'entrée ξ.

La *fonction de transition finale* $\tilde{\lambda}(a_m, \xi)$ se définit dans le modèle de Moore sur l'ensemble $A \times (E \cup \{e\})$ de la façon suivante:

$$\tilde{\lambda}(a_m, e) = \lambda(a_m);$$

$$\tilde{\lambda}(a_m, \xi) = \begin{cases} \lambda(\tilde{\delta}(a_m, \xi)) \text{ pour tout } \xi \in E \text{ si la fonction } \tilde{\delta}(a_m, \xi) \\ \quad \text{est définie;} \\ \text{non définie si la fonction } \tilde{\delta}(a_m, \xi) \text{ n'est pas définie.} \end{cases}$$

Dans le modèle de Mealy

$$\tilde{\lambda}(a_m, \xi) = \begin{cases} \lambda(\tilde{\delta}(a_m, \xi'), z_{i_k}), \ \xi = \xi' z_{i_k}, \text{ si la fonction } \tilde{\delta}(a_m, \xi') \\ \quad \text{est définie;} \\ \text{non définie si la fonction } \tilde{\delta}(a_m, \xi') \text{ n'est pas définie.} \end{cases}$$

On a ici $\xi = z_{i_1} \ldots z_{i_{k-1}} z_{i_k}$, $\xi' = z_{i_1} \ldots z_{i_{k-1}}$.

Ainsi donc, dans le modèle de Moore, $\tilde{\lambda}(a_m, \xi)$ est le signal de sortie associé à l'état final. Dans le modèle de Mealy $\tilde{\lambda}(a_m, \xi)$ est le signal de sortie élaboré à la dernière transition (transition vers l'état final). Il apparaît à la sortie à l'instant même d'arrivée de la dernière lettre du mot d'entrée ξ.

Cherchons par exemple la fonction $\tilde{\lambda}(a_1, z_1z_2z_1z_1)$ pour l'automate S_1 (voir tableaux 2-1, 2-2). On a

$$\tilde{\delta}(a_1, z_1z_2z_1z_1) = \tilde{\delta}(\delta(a_1, z_1), z_2z_1z_1) = \tilde{\delta}(\delta(a_2, z_2), z_1z_1) =$$
$$= \tilde{\delta}(\delta(a_2, z_1), z_1) = \tilde{\delta}(a_3, z_1) = \delta(a_3, z_1) = a_2.$$

Dans cette suite de transitions, la dernière est (a_3, z_1), aussi

$$\tilde{\lambda}(a_1, z_1z_2z_1z_1) = \lambda(a_3, z_1) = w_3.$$

Introduisons la fonction $\omega(a_m, \xi)$: c'est la *réponse*, ou l'*évolution de l'automate* à partir de l'état a_m lorsqu'il reçoit à son entrée la séquence $\xi = z_{i_1} \ldots z_{i_k}$. Elle se définit sur l'ensemble $A \times \times (E \cup \{e\})$ de la façon suivante.

Modèle de Mealy. La fonction $\omega(a_m, \xi)$ n'est pas définie si la fonction $\tilde{\delta}(a_m, \xi')$ ne l'est pas ; on a ici $\xi' = z_{i_1} \ldots z_{i_{k-1}}$, c'est-à-dire $\xi = \xi' z_{i_k}$. Si la fonction $\tilde{\delta}(a_m, \xi')$ est définie, on a

$$\omega(a_m, \xi) = \lambda(a_m, z_{i_1})\, \omega(\delta(a_m, z_{i_1}), z_{i_2} \ldots z_{i_k}) =$$
$$= \lambda(a_m, z_{i_1})\, \lambda(a_{i_2}, z_{i_2}) \ldots \lambda(a_{i_k}, z_{i_k}) = w_{i_1} \ldots w_{i_k}.$$

Ici $a_{i_2} = \delta(a_m, z_{i_1})$; $w_{i_j} = \lambda(a_{i_j}, z_{i_j})$; $j = 2, 3, \ldots, k$; $w_{i_1} = \lambda(a_m, z_{i_1})$.

On a, par exemple, pour l'automate S_1 :

$$\omega(a_1, z_1z_2z_1z_1) = \lambda(a_1, z_1)\, \omega(\delta(a_1, z_1), z_2z_1z_1) =$$
$$= w_1\omega(a_2, z_2z_1z_1) = w_1\lambda(a_2, z_2)\, \omega(\delta(a_2, z_2), z_1z_1) =$$
$$= w_1w_1\omega(a_2, z_1z_1) = w_1w_1\lambda(a_2, z_1)\, \omega(\delta(a_2, z_1), z_1) =$$
$$= w_1w_1w_3\lambda(a_3, z_1) = w_1w_1w_3w_3.$$

Il est facile de voir que pour l'automate partiel de Mealy S_3 (tableau 2-6) la fonction $\omega(a_2, z_1z_1z_2z_1z_2)$ est égale à $w_3w_3w_2w_3$ — ; le tiret en dernière position signifie que le dernier signal de sortie n'est pas défini dans la réponse. Par contre, la réponse elle-même est définie, car la fonction $\tilde{\delta}(a_2, z_1z_1z_2z_1)$ est définie.

Or, pour ce même automate, la fonction $\omega(a_2, z_1z_1z_2z_2z_2)$ est non définie, car la fonction $\tilde{\delta}(a_2, z_1z_1z_2z_2)$ ne l'est pas (voir plus haut). Il est possible également de définir la réponse au mot d'entrée ξ d'un automate de Mealy se trouvant en l'état a_m en fonction de la sortie finale :

$$\omega(a_m, \xi) = \tilde{\lambda}(a_m, z_{i_1})\, \tilde{\lambda}(a_m, z_{i_1}z_{i_2}) \ldots \tilde{\lambda}(a_m, z_{i_1} \ldots z_{i_k}).$$

Modèle de Moore. La fonction $\omega(a_m, \xi)$ n'est pas définie si la fonction $\tilde{\delta}(a_m, \xi)$ ne l'est pas. Si la fonction $\tilde{\delta}(a_m, \xi)$ est définie, on a

$$\omega(a_m, \xi) = \lambda(\delta(a_m, z_{i_1}))\, \omega(\delta(a_m, z_{i_1}), z_{i_2} \ldots z_{i_k}) =$$
$$= \lambda(a_{i_2})\, \lambda(a_{i_3}) \ldots \lambda(a_{i_k+1}) = w_{i_2}w_{i_3} \ldots w_{i_{k+1}}.$$

Ici

$$a_{i_2} = \delta(a_m, z_{i_1}) ;\ w_{i_j} = \lambda(a_{i_j}),\ j = 2, \ldots, k+1.$$

Cherchons par exemple la réponse $\omega(a_1, z_2z_1z_1z_2)$ de l'automate de Moore défini par la table des transitions marquée (voir tableau 2-3) : $\omega(a_1, z_2z_1z_1z_2) = \lambda(\delta(a_1, z_2))\,\omega(\delta(a_1, z_2), z_1z_1z_2) =$

$$= \lambda(a_4)\,\omega(a_4, z_1z_1z_2) = w_2\lambda(\delta(a_4, z_1))\,\omega(\delta(a_4, z_1), z_1z_2) =$$
$$= w_2\lambda(a_3)\,\omega(a_3, z_1z_2) = w_2w_3\,\lambda(\delta(a_3, z_1))\omega(\delta(a_3, z_1), z_2) =$$
$$= w_2w_3\lambda(a_5)\,\omega(a_5, z_2) = w_2w_3w_3\lambda(\delta(a_5, z_2)) =$$
$$= w_2w_3w_3\lambda(a_1) = w_2w_3w_3w_1.$$

Il est évident que dans l'automate de Moore le signal de sortie $w_{i_1} = \lambda(a_m)$ produit à l'instant i_1 ne dépend pas du signal d'entrée z_{i_1} mais seulement de l'état a_m. Cela signifie que le signal w_{i_1} cesse de dépendre, à partir de l'instant i_1, du mot d'entrée accepté à l'entrée de l'automate. On entend donc par réponse de l'automate de Moore, se trouvant dans l'état a_m et attaqué par une séquence d'entrée $\xi = z_{i_1}z_{i_2}\ldots z_{i_k}$, une séquence de sortie de même longueur $\omega(a_m, \xi) = w_{i_2}w_{i_3}\ldots w_{i_{k+1}}$ mais décalée d'un intervalle d'horloge.

Les trois fonctions sont réunies dans le tableau p. 34.

Généralisons la fonction des transitions $\delta: A \times Z \to A$ à l'ensemble $\mathfrak{A} \times Z$, où $\mathfrak{A}$ est l'ensemble des parties de A. La fonction $\delta(B, z_f)$, où $B \subseteq A$, $z_f \in Z$, se définit de la façon suivante :

$$\delta(B, z_f) = \{a_s \mid a_s = \delta(a_m, z_f),\ a_m \in B\}.$$

Autrement dit,

$$\delta(B, z_f) = \bigcup_{a_m \in B} \{\delta(a_m, z_f)\}.$$

Ainsi donc, $\delta(B, z_f)$ est l'ensemble des états auxquels l'automate attaqué par le signal d'entrée z_f peut passer à partir de n'importe quel état appartenant à B. On dit que l'ensemble d'états $\delta(B, z_f)$ est le *successeur* de l'ensemble B suivant l'entrée z_f si et seulement si

$$\delta(B, z_f) = \{a_s \mid a_s = \delta(a_m, z_f),\ a_m \in B\}.$$

Par exemple, supposons que pour l'automate S_2 (tableau 2-3) $B = \{a_1, a_4, a_5\}$. On a alors

$$\delta(B, z_1) = \{a_2, a_3\};\ \delta(B, z_2) = \{a_1, a_4\}.$$

Nous terminerons ce paragraphe en donnant la définition d'automates synchrones et asynchrones.

On dit qu'un état a_s de l'automate S est *stable* si pour toute entrée $z_f \in Z$ telle que $\delta(a_m, z_f) = a_s$ on a $\delta(a_s, z_f) = a_s$. Cela signifie

Modèle de Mealy

Mot d'entrée	z_{i_1}	z_{i_2}	...	z_{i_k}	
					$\tilde{\delta}(a_m, \xi)$
Séquence d'états internes	a_m	$a_{i_2} = \delta(a_m, z_{i_1})$	...	$a_{i_k} = \delta(a_{i_{k-1}}, z_{i_{k-1}})$	$a_{i_{k+1}} = \delta(a_{i_k}, z_{i_k})$
Réponse	$w_{i_1} = \lambda(a_m, z_{i_1})$	$w_{i_2} = \lambda(a_{i_2}, z_{i_2})$	...	$w_{i_k} = \lambda(a_{i_k}, z_{i_k})$	
				$\tilde{\lambda}(a_m, \xi)$	
	$\omega(a_m, \xi)$				

Modèle de Moore

Mot d'entrée	z_{i_1}	z_{i_2}	...	z_{i_k}	
					$\delta(a_m, \xi)$
Séquence d'états internes	a_m	$a_{i_2} = \delta(a_m, z_{i_1})$		$a_{i_k} = \delta(a_{i_{k-1}}, z_{i_{k-1}})$	$a_{i_{k+1}} = \delta(a_{i_k}, z_{i_k})$
Réponse	$w_{i_1} = \lambda(a_m)$	$w_{i_2} = \lambda(a_{i_2})$		$w_{i_k} = \lambda(a_{i_k})$	$w_{i_{k+1}} = \lambda(a_{i_{k+1}})$
					$\tilde{\lambda}(a_m, \xi)$
		$\omega(a_m, \xi)$			

qu'une fois passé à un état donné sous l'action d'un signal d'entrée z_f, l'automate ne peut en sortir qu'en recevant à son entrée un autre signal d'entrée, distinct de z_f. On dit que l'automate S est *asynchrone* si chacun de ses états $a_s \in A$ est stable. Dans le cas contraire, on dit que l'automate S est *synchrone* [60, 45].

Il est à noter que tous les automates réels (physiques) sont asynchrones ; on s'arrange toujours pour assurer la stabilité de leurs états, par exemple au moyen d'une horloge. Or, dans la théorie abstraite, où l'automate n'est qu'un modèle mathématique auquel on n'associe

Tabl au 2 7

$\delta : A \times Z \to A$,
$\lambda : A \to W$

	w_1	w_3	w_2
	a_1	a_2	a_3
z_1	a_2	a_2	a_2
z_2	a_3	a_2	a_3
z_3	a_1	a_1	a_3

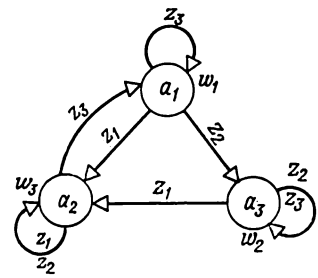


Fig. 2-6. Graphe d'un automate de Moore asynchrone

aucune technologie ni composants concrets, on peut se borner à ne considérer que les automates synchrones, ce que nous ferons précisément tout au long des chapitres 2, 3 et 4. Un exemple d'automate de Moore asynchrone est donné dans le tableau 2-7 et sur la figure 2-6 ; on remarque que tous ses états sont stables.

Il est évident que si dans la table des transitions d'un automate asynchrone un état a_s figure à l'intersection de la ligne z_f et de la colonne a_m ($m \neq s$), le même état apparaîtra sûrement dans la même ligne, à l'intersection avec la colonne a_s. Sur le graphe de l'automate asynchrone, si un état a_s peut être atteint par transition à partir d'autres états sous l'action de certains signaux, le sommet a_s présentera une boucle marquée par les symboles des mêmes signaux d'entrée. En analysant les tableaux 2-1, 2-3 et 2-4, on remarque que les automates S_1, S_2 et S_3 sont synchrones.

Assez souvent, en disant qu'un automate est synchrone ou asynchrone, on sous-entend que son circuit est synchronisé ou non par des impulsions d'horloge. On peut montrer que même en présence d'impulsions d'horloge, un automate peut être et généralement est asynchrone, au sens de la définition proposée. Il est vrai qu'il s'agit alors d'un automate structuré (voir chapitre 5) ; quant à l'automate

abstrait, il est dépourvu de sens de parler d'une synchronisation quelconque. La définition précédente d'automates synchrone et asynchrone s'étend d'ailleurs sans difficulté au niveau de la structure *).

2-3. Lien entre les modèles de Mealy et de Moore

La notion de réponse d'un automate à une séquence d'entrée donnée qui a été introduite dans le paragraphe précédent permet de donner une définition rigoureuse de l'équivalence de deux automates complètement spécifiés. On dit que deux automates S_A et S_B ayant respectivement les mêmes alphabets d'entrée et de sortie sont *équivalents* si, étant mis à l'état initial, ils présentent les mêmes réponses à un même mot d'entrée quelconque. Une définition plus développée de la notion d'équivalence des automates sera donnée dans le troisième chapitre.

Examinons les réponses de l'automate de Mealy S_1 (tableaux 2-1, 2-2, fig. 2-3) et de l'automate de Moore S_2 (tableau 2-3, fig. 2-4), se trouvant en l'état a_1, au mot d'entrée $z_1z_2z_1z_2z_2$, en étudiant leurs tables des transitions ou leurs graphes.

Automate S_1

Mot d'entrée	z_1	z_2	z_1	z_2	z_2	
Séquence d'états internes	a_1	a_2	a_2	a_3	a_1	a_3
Mot de sortie		w_1	w_1	w_3	w_1	w_2

Automate S_2

Mot d'entrée	z_1	z_2	z_1	z_2	z_2	
Séquence d'états internes	a_1	a_2	a_2	a_5	a_1	a_4
Mot de sortie	w_1	w_1	w_1	w_3	w_1	w_2

On notera que les automates S_1 et S_2 présentent la même réponse à un même mot d'entrée donné. Cela ne veut pas pour autant dire que ces deux automates sont équivalents, c'est-à-dire que, mis à l'état initial, ils produiraient les mêmes sorties sous l'action de divers mots d'entrée de longueur quelconque. La question se pose de savoir si, pour n'importe quel automate de Mealy, on peut construire un automate de Moore équivalent, et inversement. La réponse est affirmative ; nous décrirons plus tard les algorithmes de passage de l'automate de Moore à l'automate de Mealy équivalent, et vice

*) Un modèle d'automate apériodique, qui reste opérationnel en dépit des retards apportés par ses composants, a été proposé par V. Varchavski avec coll. [62]. E. Hurvitz a proposé des automates polysynchrones [29]. Les automates asynchrones ont été étudiés dans le détail par E. Jakoubaïtis [64].

versa. Dans cette description, nous négligerons, dans les automates de Moore, le signal de sortie $\lambda(a_1)$ correspondant à l'état initial.

Etudions d'abord le passage de l'automate de Moore à celui de Mealy. Soit un automate de Moore $S_A = (A_A, Z_A, W_A, \delta_A, \lambda_A, a_{1A})$, tel que $A_A = \{a_1, \ldots, a_M\}$, $Z_A = \{z_1, \ldots, z_F\}$, $W_A = \{w_1, \ldots, w_G\}$, $\delta_A : A_A \times Z_A \to A_A$, $\lambda_A : A_A \to W_A$ et $a_{1A} = a_1$, état initial. Construisons un automate de Mealy $S_B = (A_B, Z_B, W_B, \delta_B, \lambda_B, a_{1B})$ tel que

$$A_B = A_A; \quad Z_B = Z_A; \quad W_B = W_A; \quad \delta_B = \delta_A; \quad a_{1B} = a_{1A} = a_1.$$

La fonction des sorties $\lambda_B : A_B \times Z_B \to W_B$ sera définie comme suit : si dans l'automate de Moore on a $\delta_A(a_m, z_f) = a_s$ et $\lambda_A(a_s) = w_g$, on a dans l'automate de Mealy $\lambda_B(a_m, z_f) = w_g$.

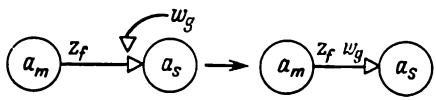


Fig. 2-7. Illustration du passage du modèle de Moore à celui de Mealy

Le passage de l'automate de Moore à celui de Mealy dans le cas de la définition graphique est illustré sur la figure 2-7 : le signal de sortie (w_g) écrit à côté du sommet (a_s) est affecté à tous les arcs incidents vers l'intérieur à ce sommet. Sur la figure 2-8 on voit le graphe de l'automate de Mealy S_4 équivalent à l'automate de Moore S_2 (fig. 2-4).

En cas de définition tabulaire, la table des transitions de S_B se confond avec celle de S_A, tandis que la table des sorties de S_B se déduit de sa table des transitions en remplaçant le symbole a_s, qui figure à l'intersection de la colonne a_m et de la ligne z_f, par le symbole du signal de sortie w_g marquant la colonne a_s de la table des transitions de S_A.

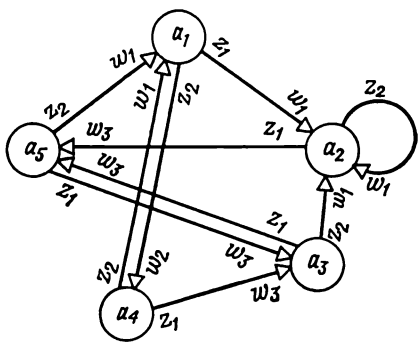
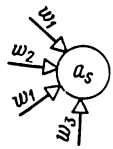


Fig. 2-8. Automate de Mealy S_4 équivalent à l'automate de Moore S_2

Le procédé de construction même de l'automate de Mealy S_B assure son équivalence à l'automate de Moore S_A. En effet, attaqué par un signal d'entrée $z_f \in Z$, l'automate S_A passera de l'état a_m à l'état $a_s = \delta_A(a_m, z_f)$ en produisant le signal de sortie $w_g = \lambda_A(a_s)$. De même, l'automate de Mealy correspondant S_B passerait lui aussi de l'état a_m à l'état a_s, car $\delta_B(a_m, z_f) = \delta_A(a_m, z_f) = a_s$, et produirait le même signal de sortie w_g, d'après le procédé de construction de la fonction λ_B. Ainsi donc, les

automates S_A et S_B présentent un comportement identique pour une séquence d'entrée de longueur 1. On montre sans peine par récurrence que n'importe quel mot d'entrée de longueur finie appliqué aux entrées des automates S_A, S_B mis en l'état a_m provoquera l'apparition des mêmes sorties, ce qui prouve que les automates S_A et S_B sont équivalents.



$A_s=\{(a_s,w_1),(a_s,w_2),(a_s,w_3)\}$

Fig. 2-9. Construction l'ensemble A_s

Avant d'examiner le passage de l'automate de Mealy à celui de Moore, nous imposerons une contrainte à l'automate de Mealy: il ne doit présenter aucun état transitoire. S'agissant d'un automate représenté sous forme d'un graphe, on entend par *état transitoire* un sommet qui ne comporte aucun arc incident vers l'intérieur mais comporte en revanche au moins un arc incident vers l'extérieur (tel est par exemple l'état a_1 sur la figure 2-5). Soit donc un automate de Mealy $S_A = (A_A, Z_A, W_A, \delta_A, \lambda_A, a_{1A})$, où $A_A = \{a_1, \ldots, a_M\}$, $Z_A = \{z_1, \ldots, z_F\}$, $W_A = \{w_1, \ldots, w_G\}$, $\delta_A: A_A \times Z_A \to A_A$, $\lambda_A: A_A \times Z_A \to W_A$ et $a_{1A} = a_1$ est l'état initial. Construisons un automate de Moore $S_B = (A_B, Z_B, W_B, \delta_B, \lambda_B, a_{1B})$ tel que

$$Z_B = Z_A; \quad W_B = W_A.$$

Pour définir A_B, nous ferons correspondre à chaque état $a_s \in A_A$ l'ensemble des couples du type (a_s, w_g), où w_g est le signal de sortie affectant l'arc incident à a_s vers l'intérieur (fig. 2-9):

$$A_s = \{(a_s, w_g) \mid \delta_A(a_m, z_f) = a_s \quad \text{et} \quad \lambda_A(a_m, z_f) = w_g\}.$$

Le nombre d'éléments de A_s est égal au nombre des différents signaux de sortie affectant les arcs incidents vers l'intérieur à l'état a_s de l'automate S_A.

L'ensemble d'états de l'automate S_B est la réunion des ensembles A_s $(s = 1, \ldots, M)$:

$$A_B = \bigcup_{s=1}^{M} A_s.$$

Les fonctions des sorties λ_B et des transitions δ_B seront définies comme suit. A chaque état de l'automate de Moore S_B, qui représente un couple du type (a_s, w_g), nous associons un signal de sortie w_g. Si l'automate de Mealy S_A admet une transition $\delta_A(a_m, z_f) = a_s$ suivie de l'élaboration d'un signal de sortie $\lambda_A(a_m, z_f) = w_h$, l'automate S_B (fig. 2-10) doit admettre une transition à partir de l'ensemble d'états A_m, associés à l'état a_m, vers l'état (a_s, w_h) sous l'action du même signal d'entrée z_f.

Comme état initial a_{1B}, on peut retenir n'importe lequel des états de A_1 engendrés par l'état initial a_1 de l'automate S_A. Rappelons qu'en comparant les réponses des automates S_A et S_B aux différents mots d'entrée, on néglige le signal de sortie à l'instant $t = 0$ lié à l'état a_{1B} de l'automate S_B.

Considérons la transformation de l'automate de Mealy S_1 de la figure 2-3 en un automate de Moore. Dans l'automate de Mealy on

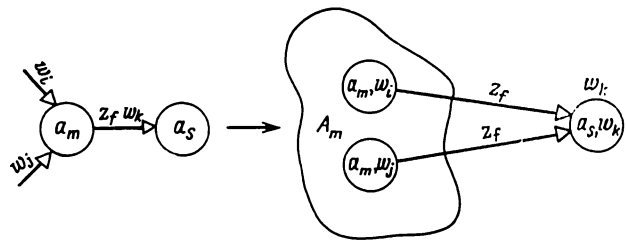


Fig. 2-10. Illustration du passage du modèle de Mealy à celui de Moore

a $Z_A = \{z_1, z_2\}$, $W_A = \{w_1, w_2, w_3\}$, $A_A = \{a_1, a_2, a_3\}$, $a_{1A} = a_1$, les fonctions δ_A et λ_A sont définies par le graphe de l'automate. Dans l'automate de Moore équivalent on a $Z_B = Z_A = \{z_1, z_2\}$, $W_B = W_A = \{w_1, w_2, w_3\}$. Construisons l'ensemble A_B. A cet effet, cherchons les ensembles de couples engendrés par chaque état de l'automate S_A *):

$$A_1 = \{(a_1, w_1)\} = \{b_1\};\ A_2 = \{(a_2, w_1), (a_2, w_3)\} = \{(b_2, b_3\};$$
$$A_3 = \{(a_3, w_2), (a_3, w_3)\} = \{b_4, b_5\}.$$

A chaque état du type (a_s, w_g), représentant un couple, nous identifions le signal de sortie w_g, second élément du couple:

$$\lambda_B(b_1) = \lambda_B(b_2) = w_1;\ \lambda_B(b_4) = w_2;\ \lambda_B(b_3) = \lambda_B(b_5) = w_3.$$

Expliquons le mode de construction de la fonction δ_B. Puisque, dans l'automate S_A, il y a, sous l'action du signal z_1, la transition de l'état a_2 à l'état a_3 suivie de l'apparition du signal w_3, l'automate S_B doit admettre une transition à partir de l'ensemble d'états $A_2 = \{b_2, b_3\}$ engendrés par a_2 vers l'état $(a_3, w_3) = b_5$ sous l'action du signal z_1. De même, il doit admettre une transition à partir des états de l'ensemble A_2 vers l'état $(a_2, w_1) = b_2$ sous l'action du signal z_2, etc. Comme état initial, on ne peut choisir que b_1, car c'est l'unique état engendré par l'état initial a_1 de l'automate S_A. Remplaçons maintenant les symboles b_i $(i = 1, \ldots, 5)$ par a_i: nous

*) Pour abréger les notations on remplace les couples par les symboles b_1, b_2, ...

retrouvons l'automate de Moore S_2 (fig. 2-4) équivalent à l'automate de Mealy S_1 de la figure 2-3.

Considérons le passage à l'automate de Moore à partir de l'automate de Mealy S_5 (fig. 2-11) qui présente un état transitoire a_1. On a comme précédemment $A_1 \cup A_2 \cup A_3 = \{(a_2, w_1), (a_2, w_2), (a_3, w_1), (a_3, w_2)\} = \{b_1, b_2, b_3, b_4\}$. Ajoutons à l'ensemble des couples obtenus — états de l'automate de Moore — l'état $(a_1, —) = b_5$ engendré par l'état transitoire a_1, en supposant que le signal de sortie pour l'état b_5 est non défini. La fonction des transitions δ_B

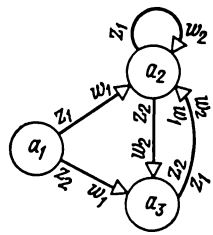


Fig. 2-11. Automate de Mealy S_5 présentant un état transitoire a_1

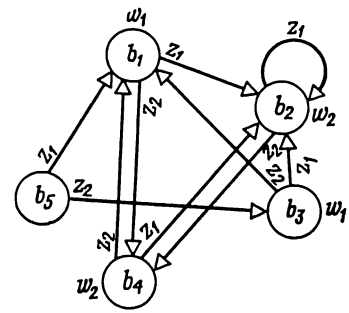


Fig. 2-12. Automate de Moore S_6 équivalent à l'automate de Mealy S_5

sera définie comme précédemment: en particulier, l'automate admettra des transitions de l'état b_5 vers $b_1 = (a_2, w_1)$ sous l'action de z_1 et vers $b_3 = (a_3, w_1)$ sous l'action de z_2. Puisque a_1 est l'état initial de l'automate de Mealy, on doit retenir comme état initial de l'automate de Moore l'état b_5 engendré par a_1. On voit sur la figure 2-12 le graphe de l'automate de Moore S_6 qui est équivalent à l'automate de Mealy de la figure 2-11.

L'équivalence des automates S_B et S_A dans la transformation d'un automate de Mealy en un automate de Moore sur l'ensemble des mots d'entrée de longueur finie se démontre facilement par récurrence, par analogie à ce qu'on a vu plus haut à l'occasion du passage inverse.

Les transformations réciproques des modèles de Mealy et de Moore exposées dans ce paragraphe montrent qu'en passant de l'automate de Moore à celui de Mealy le nombre d'états reste inchangé, alors que le passage inverse fait croître généralement le nombre d'états de l'automate de Moore (sauf si chaque état de l'automate de Mealy engendre un seul état de l'automate de Moore). En effet, si, après avoir construit l'automate de Moore S_2 (fig. 2-4) équivalent à l'automate de Mealy S_1 (fig. 2-3), on se propose de passer de nouveau à un automate de Mealy, on obtient l'automate de Mealy S_4

(fig. 2-8). Puisque la relation d'équivalence est transitive, les deux automates de Mealy S_1 et S_4 sont équivalents eux aussi, bien que S_4 possède deux états de plus que S_1. Ainsi donc, les automates équivalents peuvent avoir des nombres d'états différents, ce qui pose le problème de la recherche de l'automate minimal (c'est-à-dire ayant le plus petit nombre d'états) dans la classe d'automates équivalents. C'est E. F. Moore qui a démontré le premier l'existence d'un automate abstrait $S_{\min}$ ayant un nombre minimal d'états internes, équivalent à tout automate abstrait donné S.

2-4. Modèle composite (C-automate)

Nous appellerons C-automate abstrait le modèle mathématique d'un réseau séquentiel défini par un vecteur à huit composantes $S = (A, Z, W, U, \delta, \lambda_1, \lambda_2, a_1)$ dans lequel:

$A = \{a_1, \ldots, a_M\}$ est l'ensemble d'états internes;

$Z = \{z_1, \ldots, z_F\}$ est l'alphabet d'entrée;

$W = \{w_1, \ldots, w_G\}$ est l'alphabet de sortie du type 1;

$U = \{u_1, \ldots, u_H\}$ est l'alphabet de sortie du type 2;

$\delta: A \times Z \to A$ est la fonction des transitions réalisant l'application $D_\delta \subseteq A \times Z$ dans A;

$\lambda_1: A \times Z \to W$ est la fonction des sorties réalisant l'application $D_{\lambda_1} \subseteq A \times Z$ sur W;

$\lambda_2: A \to U$ est la fonction des sorties réalisant l'application $D_{\lambda_2} \subseteq A$ sur U;

$a_1 \in A$ est l'état initial de l'automate.

Un C-automate abstrait peut revêtir la forme d'un circuit (fig. 2-13) à une entrée, attaquée par les signaux appartenant à l'al-

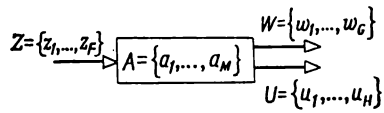


Fig. 2-13. C-automate abstrait

phabet d'entrée Z, et à deux sorties produisant des signaux qui appartiennent aux alphabets W et U. A la différence des modèles de Mealy et de Moore, un C-automate produit simultanément deux fonctions des sorties λ_1 et λ_2, caractéristiques chacune de l'un des modèles en question. Un C-automate a pour équations

$$a(t+1) = \delta(a, (t), z(t));$$
$$w(t) = \lambda_1(a(t), z(t));$$
$$u(t) = \lambda_2(a(t)), \quad t = 0, 1, 2, \ldots$$

Le signal de sortie $u_h = \lambda_2(a_m)$ est produit pendant toute la durée de l'état a_m de l'automate. Le signal de sortie $w_g =$

$= \lambda_1 (a_m, z_f)$ est produit pendant l'action du signal d'entrée z_f, l'automate placé dans l'état a_m. De toute évidence, il est facile de passer d'un C-automate à un automate de Mealy ou de Moore (compte tenu d'un décalage temporel possible d'une période d'horloge), de la même façon qu'on passe d'un automate de Mealy à un automate de Moore et inversement. Dans de nombreuses applications, on a d'ailleurs besoin des deux types de signaux de sortie. Par exemple, dans l'ordinateur certaines instructions, telles que le transfert d'informations entre registres, la progression des compteurs, etc., égalent en durée le signal d'entrée synchronisé par une impulsion d'horloge, tandis que d'autres, telles que les instructions d'accès à un additionneur combinationnel ou à un décodeur de mémoire, sont nettement plus longues, en raison du temps de fonctionnement de l'additionneur et du décodeur. Il y a donc intérêt à faire durer de tels signaux autant que dure un état de l'automate. Ainsi, des considérations d'utilité pratique justifient l'introduction du C-automate, modèle composite des automates de Mealy et de Moore.

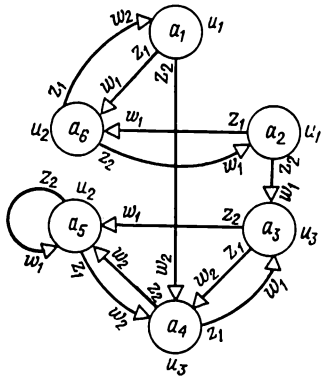


Fig. 2-14. Graphe d'un C-automate

Les C-automates, tout comme les modèles de Mealy ou de Moore, sont susceptibles d'une description tabulaire ou à l'aide des graphes. Dans le premier cas la table des transitions (tableau 2-8) est analogue à celle de l'automate de Mealy; dans la table des sorties, chaque état est marqué par un signal de sortie du type 2 qui lui correspond (tableau 2-9). Dans la description à l'aide d'un graphe, les signaux du type 1 sont écrits à côté des signaux d'entrée affectant les arcs du graphe, tandis que les signaux du type 2 le sont à côté des sommets-états correspondants. On voit sur la figure 2-14 le graphe d'un C-automate défini par les tableaux 2-8 et 2-9.

Tableau 2-8

$\delta : A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_6	a_6	a_4	a_3	a_4	a_1
z_2	a_4	a_3	a_5	a_5	a_5	a_2

Tableau 2-9

$\lambda_1 : A \times Z \to W, \ \lambda_2 : A \to U$

	u_1	u_1	u_3	u_3	u_2	u_2
	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_1	w_2	w_1	w_2	w_2
z_2	w_2	w_1	w_1	w_2	w_1	w_1

CHAPITRE 3

MINIMISATION DES AUTOMATES

3-1. Classes de compatibilité maximales

Dans ce chapitre, nous décrirons les méthodes de minimisation des automates partiels. L'intérêt majeur de ce problème réside, à notre avis, dans le fait que son énoncé mathématique — la recherche d'un recouvrement fermé minimal — permet de résoudre bon nombre de problèmes concrets, qui ne se rapportent pas tous à la théorie des automates. Tout d'abord, nous allons donner quelques définitions.

Nous dirons qu'un mot d'entrée $\xi = z_{i_1} \ldots z_{i_k}$ est *applicable* à un automate S supposé dans l'état a_m si :

1) la fonction des transitions $\delta(a_{i_j}, z_{i_j})$ est définie pour tous $j = 1, \ldots, k-1$; $a_{i_1} = a_m$, $a_{i_{j+1}} = \delta(a_{i_j}, z_{i_j})$;

2) la sortie finale $\tilde{\lambda}(a_m, \xi) = \lambda(a_{i_k}, z_{i_k})$ est définie.

Ainsi donc, la suite d'états internes correspondant à un mot d'entrée applicable et définie par la fonction des transitions ne contient aucun état non défini. C'est ainsi que pour l'automate S_1 défini par les tableaux 3-1 et 3-2 le mot d'entrée $z_3z_2z_3z_1z_4$ est appli-

Tableau 3-1

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5
z_1	a_2	a_3	a_3	—	—
z_2	—	a_5	a_4	a_1	—
z_3	a_3	a_2	—	a_2	a_1
z_4	a_2	—	a_5	—	—

Tableau 3-2

λ: $A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5
z_1	w_1	w_1	w_1	—	—
z_2	w_2	w_2	w_2	w_2	—
z_3	—	w_1	—	—	w_2
z_4	w_1	—	w_1	—	—

cable à l'état a_1 :

Mot d'entrée	$z_3 \quad z_2 \quad z_3 \quad z_1 \quad z_4$
Séquence d'états internes	$a_1 \quad a_3 \quad a_4 \quad a_2 \quad a_3 \quad a_5$
Mot de sortie	$— \quad w_2 \quad — \quad w_1 \quad w_1$

Les mots d'entrée $z_3z_4z_3z_2z_1$ et $z_4z_1z_2z_2z_3$ ne sont pas applicables à l'état a_1, car le premier mot ne définit pas la suite d'états internes :

Mot d'entrée	z_3	z_4	z_3	z_2	z_1
Séquence d'états internes	a_1	a_3	a_5	a_1	—
Mot de sortie	—	w_1	w_2	w_2	—

et le second ne définit pas la sortie finale :

Mot d'entrée	z_4	z_1	z_2	z_2	z_3	
Séquence d'états internes	a_1	a_2	a_3	a_4	a_1	a_3
Mot de sortie	w_1	w_1	w_2	w_2	—	

On dit que l'état a_m de l'automate S est recouvert par l'état a'_m de l'automate S' si tout mot d'entrée applicable à S dans l'état a_m est aussi applicable à S' dans l'état a'_m et si ce mot, appliqué à S supposé dans a_m et à S' supposé dans a'_m, provoque des réponses identiques là où les signaux de sortie de S sont définis. Par exemple, l'état b_1 de l'automate S_2 (tableaux 3-3 et 3-4) recouvre les états a_1

Tableau 3-3

δ: $B \times Z \to B$

	b_1	b_2	b_3	b_4
z_1	b_3	b_1	b_3	b_3
z_2	b_4	b_1	b_4	b_2
z_3	b_3	b_3	b_1	b_1
z_4	b_1	b_1	b_4	b_4

Tableau 3-4

λ: $B \times Z \to W$

	b_1	b_2	b_3	b_4
z_1	w_1	w_1	w_1	w_1
z_2	w_2	w_2	w_2	w_2
z_3	w_1	—	w_1	w_2
z_4	w_1	w_1	w_1	w_1

et a_2 de l'automate S_1 (tableaux 3-1 et 3-2) et l'état b_3 en recouvre les états a_2 et a_3.

L'automate S' recouvre l'automate S si chaque état a_m de S est recouvert par au moins un état a'_m de S'. Par exemple, l'état a_1 de l'automate S_1 est recouvert par les états b_1 et b_2 de l'automate S_2 ; a_2, par b_1 et b_3 ; a_3, par b_3 et b_4 ; a_4, par b_2 et b_4 ; a_5, par b_4. Ainsi donc, l'automate S_2 recouvre l'automate S_1 (ou S_1 est recouvert par S_2).

Cette définition du recouvrement n'est pas constructive en ce sens qu'étant donné un automate, elle ne permet pas de construire un automate qui le recouvre. Le lecteur ne dispose ici que d'un petit nombre de mots d'entrée pour s'assurer que notre hypothèse sur S_1 et S_2 est juste. Dans le texte qui suit, nous montrerons comment on construit S_2 en fonction de S_1 et pourquoi l'automate S_2 ainsi construit recouvre effectivement l'automate initial S_1.

Il est évident que la relation de recouvrement est réflexive et transitive mais n'est pas symétrique. On conçoit également qu'un automate S' recouvrant S « travaille » comme S dans les mêmes conditions. Autrement dit, S' « fait autant sinon davantage » que S.

Le problème de *minimisation d'un automate partiel* S consiste à chercher un automate S' qui, de tous les automates recouvrant S, admet un nombre d'états minimal *). Une contribution importante à la solution de ce problème a été apportée par M. C. Paull et S. H. Unger [50] exploitant avec profit l'idée du recouvrement fermé minimal. Introduisons les définitions nécessaires.

Deux états a_m et a_s sont dits *compatibles* (notation $a_m \sim a_s$) si, attaqué par une séquence d'entrée applicable à a_m et à a_s, l'automate produit des séquences de sortie compatibles, c'est-à-dire des séquences qui coïncident là où elles sont toutes deux définies.

Deux états a_m, a_s qui ne sont pas compatibles sont appelés *incompatibles* (notation $a_m \nsim a_s$).

Il existe une définition plus constructive de la compatibilité. On dit que deux états a_m, a_s sont compatibles si :

1) pour toute entrée $z_f \in Z$ pour laquelle les deux sorties sont définies, on a

$$\lambda\,(a_m,\ z_f) = \lambda\,(a_s,\ z_f)$$

(*compatibilité en sortie*) ;

2) pour toute entrée $z_f \in Z$ pour laquelle les deux états successeurs sont définis, on a

$$\delta\,(a_m,\ z_f) \sim \delta\,(a_s,\ z_f),$$

c'est-à-dire que les successeurs de a_m et de a_s doivent être compatibles eux aussi.

Cette définition concerne les automates de Mealy. Dans le cas des automates de Moore, la condition 1) s'énoncera ainsi : si les deux sorties, $\lambda\,(a_m)$, $\lambda\,(a_s)$, sont définies, on a $\lambda\,(a_m) = \lambda\,(a_s)$.

Il est clair que la relation de compatibilité est réflexive et symétrique mais n'est pas transitive.

Un ensemble d'états C ($C \subseteq A$) est appelé *classe de compatibilité* si tous les états appartenant à C sont compatibles deux à deux. On dit que la classe de compatibilité C est maximale si elle est plus fine que toute autre classe de compatibilité.

Soit $B \subseteq A$ une partie de l'ensemble d'états d'un automate. Rappelons qu'on dit que l'ensemble d'états C est le successeur de l'ensemble B suivant l'entrée z_f si et seulement si

$$C = \delta\,(B,\ z_f) = \{a_s \mid a_s = \delta\,(a_m,\ z_f),\ a_m \in B\}.$$

*) La première chose à faire en minimisant un automate consisterait à supprimer les états impossibles, c'est-à-dire inaccessibles à partir de l'état initial pour aucune séquence d'entrée [51].

De la définition proposée il ressort que l'ensemble des états successeurs d'une classe de compatibilité est lui aussi une classe de compatibilité.

Considérons l'algorithme de composition de couples compatibles par la méthode de la table triangulaire proposée par M. C. Paull et S. H. Unger [50]. On peut aussi trouver la description de cet algorithme dans V. Lasarev et E. Pijl [38]. Un algorithme n'utilisant pas de table triangulaire est proposé dans [36].

Tableau 3-5

Table triangulaire pour l'automate S_1

	1	2	3	4
2	2.3			
3	2.3 2.5	4.5		
4	2.3	1.5	1.4	
5	1.3	×	V	1.2

D'après les tables des transitions et des sorties de l'automate on compose une table triangulaire dont les colonnes et les lignes sont associées aux états de l'automate. Par exemple, le tableau 3-5 est la table triangulaire composée pour l'automate S_1 défini par les tableaux 3-1 et 3-2. La forme triangulaire de la table s'explique par la réflexivité et la symétrie de la relation de compatibilité des états. Pour simplifier la notation, nous mettrons i à la place de a_i.

On met donc dans la table triangulaire, à l'intersection d'une ligne m et d'une colonne s:

soit une croix (×) si les états a_m, a_s sont incompatibles suivant la sortie (tels sont, par exemple, les états a_2 et a_5, car dans le tableau 3-2, dans la ligne associée à z_3, on remarque à l'intersection avec les colonnes a_2 et a_5 des signaux de sortie différents, à savoir w_1 et w_2 respectivement);

soit l'ensemble des couples successeurs du couple (a_m, a_s) distincts de ce dernier, c'est-à-dire la totalité des couples dont la compatibilité garantit celle du couple (a_m, a_s). Par exemple, la compatibilité de (a_1, a_3) suppose celle de (a_2, a_3) et de (a_2, a_5), vu que (a_2, a_3) et (a_2, a_5) sont les successeurs de l'ensemble (a_1, a_3) suivant les sorties z_1 et z_4 respectivement (voir tableau 3-1);

soit enfin un V si le couple (a_m, a_s) n'a pas de successeurs distincts de lui-même, c'est-à-dire si le couple (a_m, a_s) est compatible sans qu'il soit nécessaire d'imposer une condition quelconque à la compatibilité d'autres couples. Dans notre exemple, c'est le cas du couple (a_3, a_5).

Pour déceler des couples d'états incompatibles (et aussi compatibles, par la même occasion), on examine la table triangulaire colonne par colonne. Tombant sur une première case $(i.j)$ marquée d'une croix (case (2.5) dans l'exemple considéré), on met une croix dans toutes les cases contenant le couple $(i.j)$ et une deuxième croix

dans la case $(i.j)$ déjà examinée. Cette procédure est renouvelée pour toutes les cases marquées d'une croix (les cases nouvellement marquées y comprises) ; elle se termine quand toutes les cases de ce type auront été épuisées. Il est évident qu'après une telle opération, les cases restées sans croix correspondent aux couples d'états compatibles, et les cases comportant une ou plusieurs croix, aux couples incompatibles. Par exemple, le tableau 3-5 soumis à l'opération décrite prend la forme du tableau 3-6. Ce dernier permet de voir que

$$1 \sim 2,\ 1 \sim 4,\ 2 \sim 3,$$
$$3 \sim 4,\ 3 \sim 5,\ 4 \sim 5;$$
$$1 \nsim 3,\ 1 \nsim 5,\ 2 \nsim 4,\ 2 \nsim 5.$$

Tableau 3-6

Table des couples compatibles construite d'après le tableau 3-5

2	2.3			
3	2.3× 2.5×	4.5		
4	2.3	1.5 × ×	1.4	
5	1.3 × ×	××	V	1.2
	1	2	3	4

Dans de nombreuses méthodes de minimisation des automates partiels, après avoir décelé tous les couples d'états compatibles, on calcule toutes les classes de compatibilité maximales. Examinons le procédé de recherche des classes de compatibilité maximales parmi les couples compatibles proposé par S. H. Unger [61].

Nous dirons qu'un ensemble d'états B $(B \subseteq A)$ est *compatible* avec un état $a_m \in A$ (notation $a_m \sim B$) si et seulement si l'on a $a_m \sim a_s$ pour tout $a_s \in B$. Dans le cas contraire on a $a_m \nsim B$.

L'algorithme de recherche de toutes les classes de compatibilité maximales se réduit à ce qui suit :

1. Commencer la liste Φ des classes de compatibilité par les couples compatibles figurant dans la colonne extrême droite de la table triangulaire comportant au moins une case sans croix. Dans notre exemple on a $\Phi = \{\overline{4.5}\}$ *).

2. Allant de droite à gauche, écrire pour chaque i-ième colonne l'ensemble d'états $A_i \sim a_i$, c'est-à-dire l'ensemble des états correspondant aux cases sans croix dans l'i-ième colonne de la table.

*) Afin de diminuer le nombre d'accolades emboîtées dans le cas où les éléments d'un ensemble sont encore des ensembles, nous utilisons les lettres barrées (surlignées), en séparant par des points les éléments de l'ensemble sous la barre. On écrira par exemple $\Phi = \{\overline{3.4.5}, \overline{2.3}, \overline{1.2}, \overline{1.4}\}$ au lieu de $\Phi = \{\{3, 4, 5\}, \{2, 3\}, \{1, 2\}, \{1, 4\}\}$. Il est clair que la notation $\Phi = \{\overline{4.5}\}$ désigne un ensemble à un élément, ou singlet, dont l'unique élément est à son tour un ensemble d'éléments 4 et 5, c'est-à-dire que $\Phi = \{\{4, 5\}\}$.

Dans notre exemple $3 \sim \overline{4.5}$ (la troisième colonne du tableau 3-6 n'a pas de croix dans les lignes 4 et 5).

Faire l'intersection de chaque A_i avec chaque terme de la liste courante Φ. On a $A_3 = \overline{4.5}$, et la liste Φ contient également un élément unique $\overline{4.5}$:

$$\overline{4.5} \cap \overline{4.5} = \overline{4.5}.$$

Si une telle intersection contient plus d'un état, ajouter sur la liste la réunion de $\{a_i\}$ avec le contenu de l'intersection:

$$\overline{3} \cup \overline{4.5} = \overline{3.4.5}\,; \quad \Phi = \{\overline{4.5}, \overline{3.4.5}\}.$$

Maximiser ensuite l'ensemble Φ obtenu, par suppression de tous les ensembles qui se rencontrent plus d'une fois sur la liste courante et de tous les ensembles qui font partie d'autres ensembles de la liste:

$$\Phi = \{\overline{3.4.5}\}.$$

Ajouter sur la liste tous les couples composés de a_i et d'un état appartenant à A_i et qui ne font pas partie d'un autre ensemble de la liste Φ (pour $i = 3$ rien à ajouter).

Voyons ce que donne le deuxième pas de l'algorithme pour la totalité des colonnes:

$$\begin{aligned} & \Phi = \{\overline{4.5}\}\,; \\ \overline{3} \sim \overline{4.5}, \quad & \Phi = \{\overline{3.4.5}\}\,; \\ \overline{2} \sim \overline{3}, \quad & \Phi = \{\overline{3.4.5}, \overline{2.3}\}\,; \\ \overline{1} \sim \overline{2.4}, \quad & \Phi = \{\overline{3.4.5}, \overline{2.3}, \overline{1.2}, \overline{1.4}\}. \end{aligned}$$

3. Ajouter sur la liste Φ, telle qu'elle se présente après le deuxième pas, tous les singlets dont l'unique élément est un ensemble qui ne figure dans aucune autre classe de compatibilité maximale (néant dans notre exemple).

Il est évident que la liste Φ ainsi dressée réunit toutes les classes de compatibilité maximales. Ainsi donc, pour l'automate S_1 (tableaux 3-1 et 3-2 et table triangulaire 3-6) l'ensemble des classes de compatibilité s'écrit

$$\Phi = \{\overline{3.4.5}, \overline{2.3}, \overline{1.2}, \overline{1.4}\}.$$

Donnons sans commentaires encore un exemple, tiré de [65]. Il s'agit de chercher toutes les classes de compatibilité maximales pour l'automate S_3 défini par les tableaux 3-7 et 3-8 (sa table triangulaire prend la forme du tableau 3-9).

Tableau 3-7

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
z_1	a_2	a_6	—	a_2	—	a_1	a_5	a_5	a_5
z_2	a_4	a_9	—	a_1	—	—	a_2	—	a_3
z_3	—	—	a_7	a_6	—	a_2	—	—	—
z_4	a_3	—	a_8	a_5	a_6	—	—	a_1	—

Tableau 3-8

λ: $A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
z_1	—	—	—	—	—	w_1	w_2	—	—
z_2	—	—	—	—	—	—	—	—	—
z_3	—	—	—	—	—	—	—	—	—
z_4	—	—	—	—	—	w_2	—	w_1	—

Tableau 3-9

Table triangulaire pour l'automate S_3

2	2.6 4.9							
3	3.8	V						
4	3.5× ×	2.6× 1.9×	6.7× 5.8×					
5	3.6	V	6.8× ×	5.6				
6	1.2	1.6	2.7	1.2 2.6	V			
7	2.5× 2.4×	5.6 2.9	V	1.2 2.5	V	××		
8	2.5 1.3	5.6	1.8	2.5 1.5	1.6	××	V	
9	2.5× 3.4×	5.9 3.9	V	2.5 1.3	V	1.5	2.3	V
	1	2	3	4	5	6	7	8

$\Phi = \{\overline{8.9}\}$;

$\bar{7} \sim \overline{8.9}$, $\Phi = \{\overline{8.9}, \overline{7.8.9}\}$ *);

$\bar{6} \sim \bar{9}$, $\Phi = \{\overline{7.8.9}, \overline{6.9}\}$;

$\bar{5} \sim \overline{6.7.8.9}$, $\Phi = \{\overline{7.8.9}, \overline{6.9}, \overline{5.7.8.9}, \overline{5.6.9}\}$;

$\bar{4} \sim \overline{5.6.7.8.9}$, $\Phi = \{\overline{5.7.8.9}, \overline{5.6.9}, \overline{4.5.7.8.9}, \overline{4.5.6.9}\}$;

$\bar{3} \sim \overline{6.7.8.9}$, $\Phi = \{\overline{4.5.7.8.9}, \overline{4.5.6.9}, \overline{3.7.8.9}, \overline{3.6.9}\}$;

$\bar{2} \sim \overline{3.5.6.7.8.9}$, $\Phi = \{\overline{4.5.7.8.9}, \overline{4.5.6.9}, \overline{3.7.8.9}, \overline{3.6.9}, \overline{2.5.7.8.9}, \overline{2.5.6.9}, \overline{2.3.7.8.9}, \overline{2.3.6.9}\}$;

$\bar{1} \sim \overline{2.3.5.6.8}$, $\Phi = \{\overline{4.5.7.8.9}, \overline{4.5.6.9}, \overline{2.5.7.8.9}, \overline{2.5.6.9}, \overline{2.3.7.8.9}, \overline{2.3.6.9}, \overline{1.5.8}, \overline{1.5.6}, \overline{1.2.5.8}, \overline{1.2.5.6}, \overline{1.2.3.8}, \overline{1.2.3.6}\}$.

Le problème de recherche de l'ensemble des classes de compatibilité maximales peut aussi se présenter comme la recherche des cliques maximales (sous-graphes complets) d'un graphe de compatibilité non orienté, dont les sommets sont associés aux états internes de l'automate initial, et les arêtes, aux couples d'états compatibles. Le graphe de la relation de compatibilité pour l'automate S_1 est tracé sur la figure 3-1.

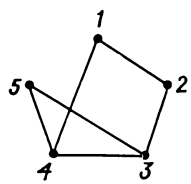


Fig. 3-1. Graphe de la relation de compatibilité pour l'automate S_1

Les exemples cités montrent que les classes de compatibilité maximales des automates partiels peuvent se rencontrer et que leur nombre peut être très supérieur à celui des états de l'automate initial. Le maximum de ce nombre est évalué à $3^{M/3}$, où M est le nombre des états de l'automate initial [46].

3-2. Recouvrement fermé minimal

Proposons-nous de fusionner les états d'une classe de compatibilité en un seul. Faisons-le par exemple dans la classe de compatibilité maximale $\overline{4.5.6.9}$ de l'automate S_3 (tableaux 3-7 et 3-8). La transformation correspondante de la table des sorties n'offre aucune difficulté. Les colonnes sont fusionnées en une colonne unique. A l'intersection de cette colonne avec une ligne z_f on met un tiret si toutes les colonnes à fusionner présentent un tiret à cette ligne. Dans le cas

*) Les ensembles supprimés au cours de la maximisation sont mis en relief par des chiffres gras.

contraire, on met dans cette ligne le signal de sortie qui figure à la ligne z_f au moins dans l'une des colonnes. Par exemple, les colonnes a_4, a_5, a_6 et a_9 donnent ensemble la colonne (écrite en ligne) w_1 — — w_2.

Comment se transforme la table des transitions ? Les successeurs de la classe $\overline{4.5.6.9}$ sont la classe $\overline{1.2.5}$ suivant l'entrée z_1, la classe $\overline{1.3}$ suivant l'entrée z_2, $\overline{2.6}$ suivant z_3 et $\overline{5.6}$ suivant z_4. Pour que l'automate résultant du fusionnement des états de $\overline{4.5.6.9}$ de l'automate S_3 soit déterministe, il faut que fusionnent les états de $\overline{1.2.5}$, $\overline{1.3}$, $\overline{2.6}$, $\overline{5.6}$. A son tour, le fusionnement des états de $\overline{1.2.5}$ nécessite celui de $\overline{2.6}$, $\overline{4.9}$, $\overline{3.6}$; de même, le fusionnement de $\overline{1.3}$ nécessite que fusionnent les états de $\overline{3.8}$, et ainsi de suite. Nous voyons que les fusionnements s'enchaînent.

On appelle *chaîne B* engendrée par un ensemble d'états C un système d'ensembles défini comme suit [25] :

1. C est considéré comme un élément de la chaîne.
2. Si Q est un élément de la chaîne, tous les successeurs de Q contenant plus d'un élément sont considérés comme des éléments de la chaîne.

On procède en outre à la maximisation de la chaîne B. Par exemple, la classe de compatibilité maximale $\overline{4.5.6.9}$ de l'automate S_3 engendre la chaîne suivante :

$$B_{\overline{4.5.6.9}} = \{\overline{4.5.6.9}, \overline{1.2.5}, \overline{1.3}, \overline{2.6}, \overline{3.6}, \overline{3.8}, \overline{1.6}, \overline{2.7}, \overline{1.8}, \overline{2.9}, \overline{3.9}\}.$$

Ici la chaîne $B_{\overline{4.5.6.9}}$ contient tous les états de l'automate initial. Tous ses éléments sont des classes compatibles (qui peuvent très bien ne pas être maximales). Le fusionnement de chaque classe en un état unique conduit à un automate déterministe qui recouvre l'automate initial mais admet 11 états : autant qu'il y a de classes dans la chaîne $B_{\overline{4.5.6.9}}$.

Ces raisonnements nous conduisent à la notion de système d'ensembles d'états fermé.

Nous disons qu'un ensemble D de classes de compatibilité est *fermé* si le successeur immédiat de chaque classe $C \in D$ est contenu au moins dans une classe de D. Il est évident que toute chaîne engendrée par une classe de compatibilité est un ensemble fermé.

Si un ensemble D de classes de compatibilité contient dans ses classes tous les états d'un ensemble A, nous dirons que D *recouvre* l'ensemble des états de l'automate initial ; l'ensemble D lui-même sera appelé *recouvrement* (un autre terme fréquemment employé est le *groupement* [51]).

Un ensemble D de classes de compatibilité s'appelle *recouvrement fermé* (*groupement régulier*) si D est un recouvrement (un groupement) et si D est fermé.

Pour s'assurer de la fermeture d'un recouvrement contenant une classe de compatibilité donnée à un élément, on n'a pas besoin de vérifier si la classe en question est fermée dans le recouvrement considéré, car chacun de ses successeurs est soit vide, soit constitué par un élément unique, ce qui revient à dire qu'il est toujours contenu dans l'une des classes constitutives du recouvrement.

Il est clair que l'ensemble des classes compatibles maximales est toujours un recouvrement fermé pour l'automate donné.

Paull et Unger [50] ont montré qu'à tout automate S' à M' états recouvrant un automate $S = (A, Z, W, \delta, \lambda, a_1)$ correspond dans S un recouvrement fermé de puissance M' ; réciproquement, à tout recouvrement fermé de puissance M' dans S correspond un automate S' qui recouvre S. De cette façon, le problème de minimisation des états d'un automate se réduit à la recherche d'un ensemble fermé de classes compatibles recouvrant l'ensemble d'états de l'automate initial et ayant le plus petit nombre de classes de compatibilité, donc à la recherche du *recouvrement fermé minimal.*

La recherche d'un automate $S' = (A', Z', W', \delta', \lambda', a_1')$, susceptible d'être représenté par le recouvrement fermé D trouvé pour l'automate initial S, ne présente pas de difficultés. A chaque classe de compatibilité $C_i \in D$ on fait correspondre un état a_i' du nouvel automate S'. Pour chaque classe $C_i \in D$ et pour chaque $z_f \in Z$, on calcule l'ensemble des successeurs T_{if} de C_i, $T_{if} = \delta(C_i, z_f)$.

Les fonctions des transitions et des sorties sont :

$$\delta'(a_i', z_f) = \begin{cases} \text{non définie si } T_{if} = \varnothing ; \\ a_j', \text{ où } a_j' \text{ est l'état associé à une des classes de compatibilité } C_j \in D \text{ contenant } T_{if} \neq \varnothing ; \end{cases}$$

$$\lambda'(a_i', z_f) = \begin{cases} \lambda(a_{i_q}, z_f) \text{ s'il existe un } a_{i_q} \in C_i \text{ tel que la fonction } \lambda(a_{i_q}, z_f) \text{ soit définie ;} \\ \text{non définie dans le cas contraire.} \end{cases}$$

Comme état initial a_1' de l'automate S', on choisit n'importe quel état dans A', à condition qu'il corresponde à une classe de compatibilité contenant l'état initial a_1 de l'automate S.

On comprend mieux à présent le mécanisme de construction de l'automate S_2 (tableaux 3-3, 3-4) à partir de l'automate S_1. On a pris comme recouvrement fermé initial pour cet automate l'ensemble des classes de compatibilité maximales $\Phi = \{\overline{1.2}, \overline{1.4}, \overline{2.3}, \overline{3.4.5}\} = \{b_1, b_2, b_3, b_4\}$.

La procédure proposée permet de voir qu'il existe plusieurs automates qui se laissent construire d'après le recouvrement fermé donné

D. En effet, il arrive que les classes de compatibilité dans D se rencontrent ; il existe donc, pour certains successeurs T_{if}, plusieurs classes $C_j \in D$ contenant T_{if}, ce qui explique l'éventualité de différentes variantes de définition de la transition δ' (a'_i, z_f). Aussi, par exemple, pour le même automate S_1 et le même ensemble d'états $\{b_1, b_2, b_3, b_4\}$ on a construit une nouvelle fonction des transitions δ'' (tabl. 3-10).

Tableau 3-10

δ'': $A' \times Z \to A'$

	b_1	b_2	b_3	b_4
z_1	b_3	b_3	b_4	b_4
z_2	b_4	b_2	b_4	b_2
z_3	b_3	b_3	b_3	b_1
z_4	b_3	b_3	b_4	b_4

Avant d'introduire deux propriétés suivantes du recouvrement fermé minimal, donnons une définition. On dit que la classe de compatibilité C_i est une classe *essentielle* dans un recouvrement fermé D si l'une des deux conditions suivantes est remplie :

1) la classe C_i est essentielle pour la condition de recouvrement, c'est-à-dire qu'il existe un état $a_s \in C_i$ tel que $a_s \notin C_j$ quel que soit $C_j \in D$, $j \neq i$;

2) si C_α est une classe essentielle de D pour la condition de recouvrement, il existe un élément $C_{\alpha t}$ de la chaîne engendrée par C_α tel que $C_{\alpha t} \subseteq C_i$ et $C_{\alpha t} \nsubseteq C_j$ quel que soit $C_j \in D$, où $j \neq i$. Dans ce cas C_i est une classe essentielle de D pour la condition de fermeture.

THÉORÈME. *Si D est un recouvrement fermé minimal, toutes ses classes sont essentielles.*

La preuve résulte immédiatement de la définition d'une classe essentielle et de la minimalité de D.

COROLLAIRE. *Aucunes deux classes distinctes d'un recouvrement fermé minimal ne sont entre elles dans une relation d'inclusion.*

3-3. Minimisation des automates partiels à l'aide des classes premières

La procédure de minimisation des automates partiels proposée par Paull et Unger [50] comprend trois étapes principales :

1. Recherche de toutes les classes de compatibilité maximales.
2. Recherche d'un recouvrement fermé minimal.
3. Reconstruction d'un nouvel automate d'après ce recouvrement fermé minimal.

Les étapes 1 et 3 sont suffisamment simples (leurs algorithmes ont été considérés dans 3-1 et 3-2). C'est à l'étape 2 que des difficultés surgissent. Il convient de noter que dans certains cas, l'ensemble

des classes compatibles maximales pour l'automate donné contient des classes qui ne font pas partie du recouvrement fermé minimal; au contraire quelques-uns de leurs sous-ensembles propres peuvent devenir éléments de ce recouvrement. Il y a en outre des cas où certaines classes de compatibilité maximales fournissent plusieurs éléments du recouvrement fermé minimal, alors que d'autres n'en fournissent pas un seul. Quant au nombre de recouvrements fermés minimaux, il s'avère quelquefois nettement supérieur au nombre des états de l'automate partiel. On comprend donc l'utilité pratique du problème de recherche d'un recouvrement fermé minimal pour l'automate donné. L'algorithme proposé dans [50] est celui d'examen complet de toutes les classes de compatibilité possibles, ainsi que des chaînes engendrées par celles-ci.

A. Grasselli et F. Luccio [25] montrent qu'en cherchant un recouvrement fermé minimal, il suffit de considérer une seule partie de l'ensemble des classes de compatibilité, à savoir le sous-ensemble des classes premières. La méthode de minimisation proposée par Grasselli et Luccio se réalise en deux étapes:

1. Construction de l'ensemble des classes premières.
2. Choix d'un recouvrement fermé minimal à partir des classes premières à l'aide d'une table particulière.

Pour définir la notion de classe première, nous introduirons deux notions préliminaires.

Nous appellerons *ensemble engendré* par une classe de compatibilité C_α l'ensemble P_α des successeurs $C_{\alpha i}$ de la classe C_α, c'est-à-dire des classes telles que

1) $C_{\alpha i} \nsubseteq C_\alpha$;
2) $C_{\alpha i} \nsubseteq C_{\alpha j}$ si $C_{\alpha j} \in P_\alpha$;
3) $C_{\alpha i}$ contient plus d'un élément.

De toute évidence, si l'on veut que la classe de compatibilité C_α vérifie la condition de fermeture sur tout ensemble de classes D, chacun des éléments de P_α doit être contenu dans une classe de D distincte de C_α. Les ensembles engendrés par les classes compatibles maximales de l'automate S_1 sont résumés dans le tableau 3-11.

Nous dirons qu'une classe compatible C_j est *exclue* par la classe compatible C_i si $C_j \subset C_i$ et $P_i \subseteq P_j$, où P_i et P_j sont des ensembles engendrés pour C_i et C_j respectivement.

Par *classe première* on entend une classe de compatibilité qui n'est exclue par aucune autre classe de compatibilité. Il est évident que toute classe de compatibilité maximale est première, car elle n'est jamais plus fine qu'une autre classe de compatibilité.

Les auteurs de [25] démontrent le théorème suivant: *il existe au moins un recouvrement fermé minimal constitué par des classes premières.*

On obtient les classes premières en décomposant chaque classe de compatibilité maximale en sous-classes et en cherchant des sous-

classes premières. Remarquons que seules se laissent décomposer les classes à ensembles engendrés non vides: en effet, pour une classe à ensemble engendré vide, l'inclusion stricte d'aucune classe première n'est possible.

On commence par dresser la liste des classes premières. La liste débute par les classes maximales ayant le plus grand nombre d'éléments. Supposons qu'elles contiennent n éléments chacune. Dans notre exemple, on choisit la classe $C_1 = \overline{3.4.5}$ du tableau 3-11.

Tableau 3-11

Ensembles engendrés par les classes maximales de l'automate S_1

Classes maximales	Ensembles engendrés
$\overline{3.4.5}$	$\overline{1.4}$, $\overline{1.2}$
$\overline{2.3}$	$\overline{4.5}$
$\overline{1.2}$	$\overline{2.3}$
$\overline{1.4}$	$\overline{2.3}$

En première itération sont générées toutes les sous-classes d'ordre $n-1$ (c'est-à-dire qui contiennent $n-1$ éléments) et l'on calcule les ensembles engendrés par ces sous-classes. Seules sont ajoutées sur la liste celles qui ne sont exclues par aucune classe première déjà figurant sur la liste. Dans notre exemple, on ajoute ceci:

$$C_2 = \overline{3.4}, \quad P_2 = \{\overline{1.4}\};$$
$$C_3 = \overline{3.5}, \quad P_3 = \varnothing;$$
$$C_4 = \overline{4.5}, \quad P_4 = \{\overline{1.2}\}.$$

Ensuite on ajoute sur la liste toutes les classes de compatibilité maximales d'ordre $n-1$:

$$C_5 = \overline{2.3}, \quad P_5 = \{\overline{4.5}\};$$
$$C_6 = \overline{1.2}, \quad P_6 = \{\overline{2.3}\};$$
$$C_7 = \overline{1.4}, \quad P_7 = \{\overline{2.3}\}.$$

Tableau 3-12

Classes premières de l'automate S_1

Classes premières	Ensembles engendrés
$C_1 = \overline{3.4.5}$	$\overline{1.4}$, $\overline{1.2}$
$C_2 = \overline{3.4}$	$\overline{1.4}$
$C_3 = \overline{3.5}$	$\varnothing$
$C_4 = \overline{4.5}$	$\overline{1.2}$
$C_5 = \overline{2.3}$	$\overline{4.5}$
$C_6 = \overline{1.2}$	$\overline{2.3}$
$C_7 = \overline{1.4}$	$\overline{2.3}$
$C_8 = \overline{4}$	$\varnothing$
$C_9 = \overline{2}$	$\varnothing$
$C_{10} = \overline{1}$	$\varnothing$

En deuxième itération sont générées toutes les sous-classes d'ordre $n-2$. Les classes génératrices sont alors les classes de compatibilité maximales d'ordre n et $n-1$. D'itération en itération on produit les sous-classes de tous les ordres. A chaque itération on retient seules les sous-classes non exclues par celles qui y figurent déjà, ainsi que les classes de compatibilité maximales de même ordre.

La liste exhaustive des classes premières de l'automate S_1 est donnée dans le tableau 3-12.

La deuxième étape de la méthode de Grasselli-Luccio consiste à chercher un

ensemble de classes premières tel qu'il soit fermé, constitue un recouvrement et contienne le plus petit nombre de classes.

A cet effet, on construit une table dite *de recouvrement et de fermeture* (voir tableau 3-13 pour l'automate S_1), constituée de deux

Tableau 3-13

Table de recouvrement et de fermeture pour l'automate S_1

	1	2	3	4	5	$\overline{1.4}$	$\overline{1.2}$	$\overline{4.5}$	$\overline{2.3}$
$C_1=\overline{3.4.5}$			×	×	×	○	○	×	
$C_2=\overline{3.4}$			×	×		○			
$C_3=\overline{3.5}$			×		×				
$C_4=\overline{4.5}$				×	×		○	×	
$C_5=\overline{2.3}$		×	×					○	×
$C_6=\overline{1.2}$	×	×					×		○
$C_7=\overline{1.4}$	×			×		×			○
$C_8=\overline{4}$				×					
$C_9=\overline{2}$		×							
$C_{10}=\overline{1}$	×								

parties. La table comporte autant de lignes qu'il y a de classes premières sur la liste. Les colonnes de la partie gauche sont les états internes de l'automate. A l'intersection de la ligne C_i avec une colonne j, on met une croix (×) dans le cas où l'état a_j appartient à la classe C_i. Cette partie est analogue au tableau des implicants premiers de Quine [53] utilisé pour minimiser les fonctions booléennes. Les ensembles de classes premières tels qu'ils aient au moins une croix dans chaque colonne de gauche de la table sont des recouvre-

ments. Un recouvrement n'est fermé que s'il vérifie les conditions spécifiées dans la partie droite de la table.

Les colonnes de droite sont composées de la façon suivante. Soient C_α une classe première, et $P_\alpha = \{C_{\alpha 1}, \ldots, C_{\alpha h}, \ldots, C_{\alpha H_\alpha}\}$ son ensemble engendré. Une colonne est réservée à chaque $C_{\alpha h} \in P_\alpha$; à l'intersection de cette colonne avec la ligne C_α, on met un rond ○. On met une croix × dans la colonne $C_{\alpha h}$ à l'intersection avec les lignes correspondant aux classes premières qui contiennent $C_{\alpha h}$. Il est clair que si une ligne appartenant à un recouvrement fermé contient un ○ à l'intersection avec une colonne quelconque, l'ensemble des lignes constituant ce recouvrement fermé doit nécessairement comprendre au moins une ligne qui présente une croix × dans cette même colonne.

Grasselli et Luccio [25] suggèrent des règles de réduction de la table, grâce auxquelles on obtient une table cyclique de recouvrement et de fermeture. La recherche d'un recouvrement fermé minimal à l'aide de cette table est facilitée par les méthodes de programmation linéaire en nombres entiers. Beaucoup d'auteurs ont proposé leurs méthodes de réduction des tables de ce type ; nous ne les décrirons pas ici en raison de leur extrême complexité *).

W. S. Meisel [43] propose une méthode de recherche d'un recouvrement fermé minimal constitué uniquement de classes premières. Le procédé consiste à tracer des arbres et à faire correspondre les classes premières à leurs sommets. Nous illustrerons cette méthode sur l'exemple de l'automate S_4 (tableaux 3-14, 3-15) dont les classes premières sont résumées dans le tableau 3-16. L'automate S_4 est tiré du travail [25] de Grasselli et Luccio. Le même exemple est utilisé — avec quelques erreurs — par Meisel dans [43].

Les arbres sont tracés sur la figure 3-2. Ce sont des structures ramifiées générées de gauche à droite. La classe première qui se trouve à l'extrême gauche constitue l'élément initial de génération d'un arbre : c'est sa racine. Sur la figure 3-2, les éléments initiaux sont les classes premières $\overline{1.2.4.5}$ et $\overline{1.7}$.

L'algorithme de traçage des arbres se réduit aux opérations suivantes.

1. Choisir un état de l'automate contenu dans le plus petit nombre de classes premières. (S'il existe plus d'un état de ce type, on arrête le choix sur l'un quelconque d'eux.) Toutes les classes premières contenant cet état seront les éléments initiaux des arbres à générer.

---

*) Leur difficulté tient non seulement à ce que la recherche d'un recouvrement fermé minimal d'après la table correspondante n'est pas chose facile, mais aussi au fait que les classes premières sont trop nombreuses. Par exemple, l'automate S_3 à neuf états (tableaux 3-7, 3-8) admet soixante-dix classes premières.

Tableau 3-14

δ: $A \times Z \rightarrow A$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	a_1	a_2	a_2	—	a_2	a_2	—	a_1
z_2	—	a_4	a_4	a_5	a_5	a_3	a_3	a_5
z_3	a_4	a_1	a_1	—	a_1	—	—	a_4
z_4	a_5	—	—	a_2	—	a_8	a_5	a_2
z_5	a_2	a_1	—	a_2	a_2	a_6	—	a_2
z_6	a_1	a_1	—	—	a_5	a_7	a_7	a_5
z_7	—	—	a_7	a_1	a_1	—	a_6	a_1

Tableau 3-15

λ: $A \times Z \rightarrow W$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8
z_1	w_1	w_1	w_1	—	—	w_1	—	w_2
z_2	—	w_2	w_2	—	—	—	w_2	w_1
z_3	w_1	—	w_2	—	—	w_2	—	w_2
z_4	w_2	—	—	—	—	w_2	w_2	w_1
z_5	w_1	—	—	w_1	—	w_2	—	—
z_6	—	w_2	—	—	—	w_1	w_1	—
z_7	—	—	w_1	—	w_2	—	w_1	w_2

Pour l'automate S_4, tels sont les états a_1, a_5, a_8 contenus chacun dans deux classes premières. Nous choisirons a_1 : cela revient à choisir comme eiements initiaux les classes premières $\overline{1.2.4.5}$ et $\overline{1.7}$.

2. Tracer un arbre à partir de chaque racine de la façon suivante.

a) On retient comme sommet de l'arbre une classe première C_i telle que l'ensemble engendré par elle n'est pas vide et n'est pas recouvert par l'ensemble D (C_i) des classes premières se trouvant sur le chemin qui passe par C_i (autrement dit, la classe C_i n'est pas fermée sur l'ensemble en question). De C_i sont issus autant de chemins qu'il y a de combinaisons de classes premières vérifiant la condition de fermeture de C_i.

Considérons par exemple le chemin $(\overline{1.2.4.5}, \overline{4.8}, \overline{3.6.7})$ sur la figure 3-2. Quand on ajoute le sommet $\overline{3.6.7}$, aucune des conditions de fermeture du chemin en question, $\overline{3.4}$ ou $\overline{5.8}$, n'est encore vérifiée.

Ce sont les ensembles $\{\overline{4.5.8}, \overline{2.3.4}\}$ et $\{\overline{4.5.8}, \overline{3.4}\}$ qui vérifient ses conditions de fermeture : on les retient donc pour tracer les deux chemins issus de $\overline{3.6.7}$.

b) On retient comme sommet de l'arbre une classe première C_i telle que l'ensemble engendré par elle est vide ou recouvert par l'ensemble $D\ (C_i)$ des classes premières se trouvant sur le chemin issu de C_i. Si chaque état de l'automate est contenu dans une classe de $D\ (C_i)$, l'ensemble $D\ (C_i)$ est un recouvrement fermé, minimal de surcroît si le chemin correspondant est le plus court. S'il existe des états non contenus dans les classes premières de $D\ (C_i)$, on arrête son choix sur l'état qui est contenu dans le plus petit nombre de classes premières. Chaque classe première contenant cet état détermine une branche partant de C_i.

Tableau 3-16

Classes premières de l'automate S_4

Classes premières	Ensembles engendrés
$C_1=\overline{1.2.4.5}$	$\varnothing$
$C_2=\overline{2.3.4}$	$\{\overline{1.2}, \overline{1.7}, \overline{4.5}\}$
$C_3=\overline{3.6.7}$	$\{\overline{3.4}, \overline{5.8}\}$
$C_4=\overline{4.5.8}$	$\{\overline{1.2}, \overline{1.4}\}$
$C_5=\overline{2.3}$	$\varnothing$
$C_6=\overline{3.4}$	$\{\overline{1.7}, \overline{4.5}\}$
$C_7=\overline{3.6}$	$\{\overline{3.4}\}$
$C_8=\overline{3.7}$	$\{\overline{3.4}, \overline{6.7}\}$
$C_9=\overline{6.7}$	$\{\overline{5.8}\}$
$C_{10}=\overline{4.8}$	$\varnothing$
$C_{11}=\overline{1.7}$	$\varnothing$
$C_{12}=\overline{6}$	$\varnothing$

Considérons le chemin $(\overline{1.2.4.5}, \overline{4.5.8})$ sur la figure 3-2. Chaque élément de l'ensemble $\{\overline{1.2}, \overline{1.4}\}$ engendré par la classe première $\overline{4.5.8}$ est recouvert par la classe première $\overline{1.2.4.5}$, ce qui amène à analyser l'ensemble $\{\overline{1.2.4.5}, \overline{4.5.8}\}$ qui correspond au chemin parcouru. Aucun des états a_3, a_6, a_7 n'est contenu dans les classes $\overline{1.2.4.5}$ et $\overline{4.5.8}$. Les états a_6 et a_7 sont contenus dans quatre classes premières ; l'état a_3, dans six. Nous choisissons l'état a_6 et nous traçons à partir de $\overline{4.5.8}$ les quatre branches $\overline{6}$, $\overline{3.6}$, $\overline{3.6.7}$, $\overline{6.7}$ (l'ensemble des classes premières contenant a_6).

c) Tous les chemins sont tracés parallèlement, jusqu'à ce qu'on obtienne un premier recouvrement fermé minimal : par ce procédé on se ménage la peine de prolonger l'arbre au-delà du niveau atteint. Puisque le chemin $(\overline{1.2.4.5}, \overline{4.5.8}, \overline{6.7}, \overline{2.3})$ détermine déjà un recouvrement fermé minimal, il est inutile de tracer des chemins de longueur supérieure à 4.

L'algorithme proposé est facile à justifier. En effet, pour être retenue comme une branche, la classe première examinée doit vérifier les conditions de recouvrement et de fermeture. Or, pour que les conditions de recouvrement soient vérifiées, au moins un élément initial doit nécessairement appartenir à un recouvrement fermé minimal.

REMARQUE. La méthode de [43] conduit souvent à des arbres redondants, car on n'y propose aucun critère qui permettrait de s'arrêter à bon moment ou d'éliminer les chemins superflus.

Considérons par exemple le chemin $(\overline{1.2.4.5}, \overline{4.5.8}, \overline{6}, \overline{3.6.7})$ sur la figure 3-2. Dans ce chemin $\overline{6} \subset \overline{3.6.7}$; or, en vertu du corollaire

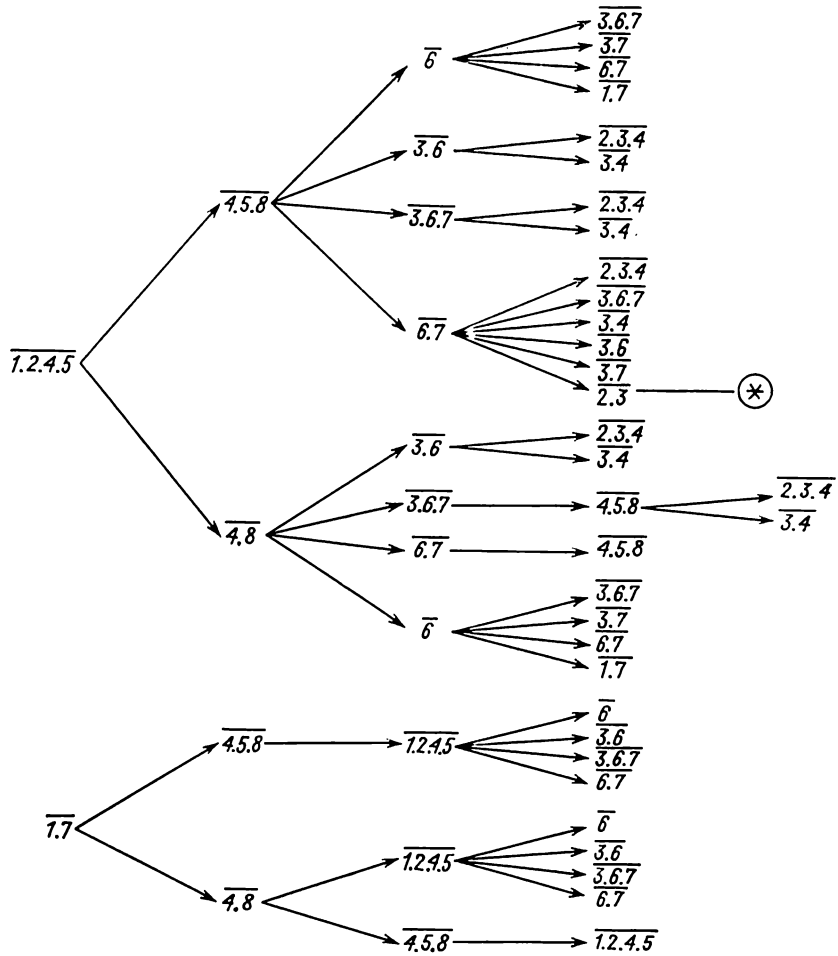


Fig. 3-2. Arbres des recouvrements de l'automate S_4

cité en fin du § 3-2, ce chemin peut être supprimé, car dans un recouvrement fermé minimal aucunes deux classes ne peuvent être entre elles dans une relation d'inclusion. Pour la même raison, on est en

droit de ne pas considérer les chemins $(\overline{1.2.4.5}, \overline{4.5.8}, \overline{6}, \overline{6.7})$, $(\overline{1.2.4.5}, \overline{4.5.8}, \overline{6.7}, \overline{3.6.7})$, $(\overline{1.2.4.5}, \overline{4.8}, \overline{6.7}, \overline{4.5.8})$, etc. Ce critère d'élimination permet souvent de réduire sensiblement le nombre de branches.

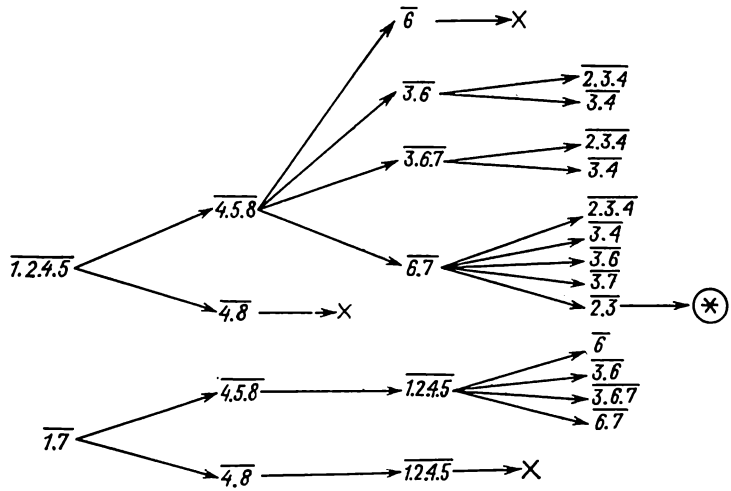


Fig. 3-3. Arbres réduits des recouvrements de l'automate S_4

Compte tenu de cette remarque, on peut simplifier la minimisation de l'automate S_4: on s'en rend compte en confrontant les figures 3-2 et 3-3. La minimisation de S_1 (tableaux 3-1, 3-2) est illustrée

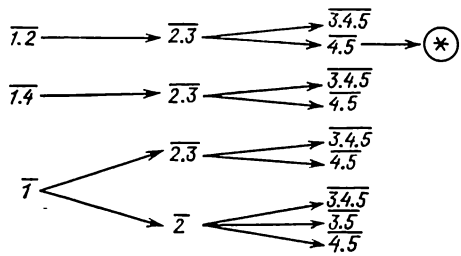


Fig. 3-4. Minimisation de l'automate S_1

sur la figure 3-4 (les classes premières de S_1 sont résumées dans le tableau 3-12). Pour S_1, un recouvrement fermé minimal sera $A' = \{\overline{1.2}, \overline{2.3}, \overline{4.5}\}$. L'automate minimal correspondant à ce recouvrement est défini par les tableaux 3-17 et 3-18.

Tableau 3-17

δ': $A' \times Z \to A'$

	$\overline{1.2}$	$\overline{2.3}$	$\overline{4.5}$
z_1	$\overline{2.3}$	$\overline{2.3}$	—
z_2	$\overline{4.5}$	$\overline{4.5}$	$\overline{1.2}$
z_3	$\overline{2.3}$	$\overline{1.2}$	$\overline{1.2}$
z_4	$\overline{1.2}$	$\overline{4.5}$	—

Tableau 3-18

λ': $A' \times Z \to W$

	$\overline{1.2}$	$\overline{2.3}$	$\overline{4.5}$
z_1	w_1	w_1	—
z_2	w_2	w_2	w_2
z_3	w_1	w_1	w_2
z_4	w_1	w_1	—

Puisque souvent il suffit de trouver au moins un recouvrement fermé minimal, nous proposerons encore une règle efficace de suppression des chemins inutiles.

Soient $R = (C_{i1}, \ldots, C_{in}, \ldots, C_{iN})$ et $Q = (C_{j1}, \ldots \ldots, C_{jm}, \ldots, C_{jM})$ deux chemins tracés suivant l'algorithme décrit ci-dessus (C_{i1} et C_{j1} sont des éléments initiaux, avec éventuellement $C_{i1} = C_{j1}$), tels que:

1) pour tout C_{jm} il existe une classe C_{in} telle que $C_{jm} \subseteq C_{in}$;

2) C_{iN} est une classe première vérifiant la condition (2b) de l'algorithme ci-dessus, c'est-à-dire que le chemin R définit un ensemble fermé;

3) $N \leqslant M$.

Dans ces conditions, le chemin R exclut le chemin Q. On marque ce dernier d'une croix (fig. 3-3) et on n'y revient plus.

Remarquons qu'en appliquant cette règle, on risque de manquer quelques recouvrements fermés minimaux; en revanche, elle garantit la construction d'au moins un recouvrement de ce type.

Compte tenu des suggestions précédentes, la méthode de recherche d'un recouvrement fermé minimal constitué par des classes premières proposée dans [43] nous semble nettement plus avantageuse que la méthode de [25], surtout en cas de calcul manuel.

3-4. Une méthode approchée de minimisation des automates partiels

Dans la minimisation des automates partiels par des méthodes exactes, les calculs deviennent vite trop volumineux lorsque le nombre d'états et de signaux d'entrée est élevé. On comprend donc l'intérêt des méthodes approchées qui permettent d'obtenir sur ordinateur une solution assez voisine de l'optimale pour des automates suffisamment compliqués. Bon nombre de méthodes approchées et la plupart des méthodes exactes nécessitent la recherche de la totalité des classes de compatibilité maximales, dont le nombre,

comme on l'a souligné, croît rapidement avec l'augmentation du nombre des états de l'automate. Dans ce paragraphe, nous étudierons dans le détail un algorithme de minimisation approché où l'on n'a pas besoin de connaître toutes les classes de compatibilité maximales et qui permet d'obtenir, sans examen exhaustif, un recouvrement fermé suffisamment proche d'un recouvrement minimal *) [52, 69].

Rappelons que toute classe de compatibilité appartenant à un recouvrement fermé est un ensemble d'états où tous les états sont compatibles deux à deux. Nous construirons donc le recouvrement fermé sous forme d'un ensemble de classes de compatibilité recouvrant l'ensemble des états de l'automate, tel que chaque ensemble de successeurs de chaque classe soit contenu dans une classe quelconque de cet ensemble.

Soit D l'ensemble maximisé des classes de compatibilité pour un automate partiel S: cela veut dire qu'aucune classe de D ne fait partie d'une autre classe de D. Supposons que D n'est pas un recouvrement fermé. Pour en faire un recouvrement fermé, nous étendrons les classes de compatibilité de D ou nous y ajouterons d'autres classes. Composons un ensemble Q avec des ensembles engendrés par les classes de D ou avec des singlets $\{a_m\}$, où a_m est un état non contenu dans les éléments de D ou de Q. De cette façon, la réunion $D \cup Q$ recouvre l'ensemble des états de l'automate initial. Il est clair que si $Q = \varnothing$, alors D est un recouvrement fermé.

La transformation de D en un recouvrement fermé est une procédure itérative: à chaque pas on essaie d'inclure un élément $q_t \in Q$ dans une classe quelconque de D. Si D offre plus d'une classe admettant q_t, on choisit celle qui minimise, dans Q, le nombre d'ensembles engendrés, c'est-à-dire d'éléments de Q contenant plus d'un état. Si l'inclusion tentée s'avère impossible à cause de l'incompatibilité entre un état quelconque de q_t et les états de toutes les classes de D, on ajoute q_t à D.

Comme D, on choisit initialement l'ensemble contenant le plus grand nombre d'éléments: $D = \{\{a_i\}, \ldots, \{a_j\}, \ldots, \{a_k\}\}$, où $a_i, \ldots, a_j, \ldots, a_k$ sont des états incompatibles deux à deux. La puissance (nombre d'états) de cet ensemble est la borne inférieure du nombre d'états de l'automate minimal recouvrant l'automate donné. Pour construire l'ensemble initial D, on peut chercher les classes d'incompatibilité maximales d'après la table triangulaire (par analogie à la recherche des classes de compatibilité maximales) et choisir parmi elles la classe contenant le plus grand nombre d'éléments. On aboutit au même résultat en traçant le graphe de la relation d'incompatibilité et en cherchant le sous-graphe complet ayant le plus grand nombre de sommets. D'autre part, un choix stochasti-

*) G. Friznovitch [17] a proposé une approche intéressante du problème de minimisation des automates, basée sur le coloriage des sommets des graphes.

que de l'ensemble initial d'états incompatibles deux à deux s'avère suffisamment efficace.

Exemple. Appliquons cet algorithme à la recherche d'un recouvrement fermé pour l'automate S_4 défini par les tableaux 3-14 et

Tableau 3-19

Table triangulaire de l'automate S_4

2	1.4						
3	××	V					
4	2.5	1.2 4.5	1.7 4.5				
5	1.2 1.4	1.2, 4.5 1.5	××	V			
6	××	××	3.4	××	2.6, 5.7 3.5 ××		
7	V	××	3.4 6.7	1.6, 3.5 × 2.5 ×	××	5.8	
8	××	××	××	V	1.2 1.4	××	××
	1	2	3	4	5	6	7

3-15. Sa table triangulaire est présentée par le tableau 3-19. Cherchons l'ensemble des classes d'incompatibilité maximales Φ_{inc}:

$$\Phi_{\text{inc}} = \{\overline{7.8}\};$$

$$\overline{6} \not\sim \overline{8}, \quad \Phi_{\text{inc}} = \{\overline{7.8}, \overline{6.8}\};$$

$$\overline{5} \not\sim \overline{6.7}, \quad \Phi_{\text{inc}} = \{\overline{7.8}, \overline{6.8}, \overline{5.6}, \overline{5.7}\};$$

$$\overline{4} \not\sim \overline{6.7}, \quad \Phi_{\text{inc}} = \{\overline{7.8}, \overline{6.8}, \overline{5.6}, \overline{5.7}, \overline{4.6}, \overline{4.7}\};$$

$$\overline{3} \not\sim \overline{5.8}, \quad \Phi_{\text{inc}} = \{\overline{7.8}, \overline{6.8}, \overline{5.6}, \overline{5.7}, \overline{4.6}, \overline{4.7}, \overline{3.5}, \overline{3.8}\};$$

$$\overline{2} \not\sim \overline{6.7.8}, \quad \Phi_{\text{inc}} = \{\overline{2.7.8}, \overline{2.6.8}, \overline{5.6}, \overline{5.7}, \overline{4.6}, \overline{4.7}, \overline{3.5}, \overline{3.8}\};$$

$$\overline{1} \not\sim \overline{3.6.8}, \quad \Phi_{\text{inc}} = \{\overline{2.7.8}, \overline{2.6.8}, \overline{1.6.8}, \overline{5.6}, \overline{5.7}, \overline{4.6}, \overline{4.7}, \overline{3.5}, \overline{1.3.8}\}.$$

Etablissons-nous sur $D = \{\overline{1}, \overline{3}, \overline{8}\}$. On a alors comme ensemble initial $Q = \{\overline{2}, \overline{4}, \overline{5}, \overline{6}, \overline{7}\}$. La procédure de construction du recouvrement fermé est résumée dans le tableau 3-20.

Tableau 3-20

Construction d'un recouvrement fermé pour l'automate S_4

Itération	D	Q
0	$\overline{1}, \overline{3}, \overline{8}$	$\overline{2}, \overline{4}, \overline{5}, \overline{6}, \overline{7}$
1	$\overline{1.7}, \overline{3}, \overline{8}$	$\overline{2}, \overline{4}, \overline{5}, \overline{6}$
2	$\overline{1.7}, \overline{2.3}, \overline{8}$	$\overline{4}, \overline{5}, \overline{6}$
3	$\overline{1.7}, \overline{2.3}, \overline{4.8}$	$\overline{5}, \overline{6}$
4	$\overline{1.7}, \overline{2.3}, \overline{4.5.8}$	$\overline{6}, \overline{1.2}, \overline{1.4}$
5	$\overline{1.7}, \overline{2.3}, \overline{4.5.8}, \overline{1.2}$	$\overline{6}, \overline{1.4}$
6	$\overline{1.7}, \overline{2.3}, \overline{4.5.8}, \overline{1.2.4}$	$\overline{6}, \overline{2.5}$
7	$\overline{1.7}, \overline{2.3}, \overline{4.5.8}, \overline{1.2.4.5}$	$\overline{6}$
8	$\overline{1.7}, \overline{2.3}, \overline{4.5.8}, \overline{1.2.4.5}, \overline{6}$	$\varnothing$

Après la troisième itération on a $D = \{\overline{1.7}, \overline{2.3}, \overline{4.8}\}$, $Q = \{\overline{5}, \overline{6}\}$. Essayons de recoller les éléments de Q à ceux de D. Puisque $\overline{5} \not\sim \overline{1.7}$, $\overline{6} \not\sim \overline{1.7}$, $\overline{5} \not\sim \overline{2.3}$, $\overline{6} \not\sim \overline{2.3}$, $\overline{6} \not\sim \overline{4.8}$ et $\overline{5} \not\sim \overline{4.8}$, on peut en quatrième itération inclure $\overline{5}$ dans la classe $\overline{4.8}$: on obtient alors $D = \{\overline{1.7}, \overline{2.3}, \overline{4.5.8}\}$. La classe $\overline{4.5.8}$ engendrera l'ensemble $\{\overline{1.2}, \overline{1.4}\}$ dont les éléments seront inclus dans Q: cela donne $Q = \{\overline{6}, \overline{1.2}, \overline{1.4}\}$. En cinquième itération, aucun élément de Q ne se recolle à aucun élément de D : on porte donc de Q à D l'ensemble $\overline{1.2}$, et ainsi de suite. Après la septième itération, on obtient un recouvrement fermé $D = \{\overline{1.7}, \overline{2.3}, \overline{4.5.8}, \overline{1.2.4.5}, \overline{6}\}$ dont le nombre d'éléments n'est que d'une unité supérieur à celui du recouvrement fermé minimal.

Le dépouillement statistique des résultats d'application de l'algorithme approché est présenté dans [52].

3-5. Minimisation des automates complètement spécifiés

Quand il s'agit de minimiser un automate complètement spécifié, au lieu de la relation de compatibilité entre états on envisage la relation d'équivalence. Deux états a_m et a_s sont dits *équivalents*

(notation $a_m \equiv a_s$) si l'on a $\omega(a_m, \xi) = \omega(a_s, \xi)$ pour n'importe quel mot d'entrée ξ; autrement dit, dans des états équivalents, l'automate présente la même réponse à n'importe quel mot d'entrée.

Deux automates $S = (A, Z, W, \delta, \lambda, a_1)$ et $S' = (A', Z, W, \delta', \lambda', a_1')$ de même type (de Mealy ou de Moore) sont *équivalents* si et seulement si pour tout état $a_m \in A$ il existe son équivalent $a_s' \in A'$, et réciproquement, si pour tout $a_s' \in A'$ il existe son équivalent $a_m \in A$.

L'automate S est minimal si et seulement si de $a_m \equiv a_s$ il ressort que $a_m = a_s$.

L'ensemble des classes maximales se transforme, dans le cas d'un automate complètement spécifié, en partition π de l'ensemble de ses états en classes d'équivalence disjointes. La partition π permet de chercher les éléments redondants dans l'ensemble d'états A. Supposons que les états a_m et a_s de l'automate soient équivalents: cela revient à dire que l'application d'une même séquence d'entrée produira toujours l'apparition d'une même séquence de sortie, qu'il soit dans a_m ou dans a_s, et ceci pour toute séquence d'entrée, par conséquent, l'un de ces deux états, par exemple a_s, peut être supprimé dans A. Si chaque classe d'équivalence ne contient qu'un seul état, l'ensemble A est *irréductible*. Par contre, si une ou plusieurs classes contiennent plus d'un élément, on est en droit d'exclure de A tous les éléments sauf un par classe: on obtient ainsi un automate ayant un nombre minimal d'états.

Cherchons la partition π pour l'automate S_5 (tableaux 3-21, 3-22) d'après sa table triangulaire (tableau 3-23):

$$\Phi = \{\overline{5.6}\};$$

$$\overline{3} \sim \overline{4},\ \Phi = \{\overline{5.6}, \overline{3.4}\};$$

$$\overline{1} \sim \overline{2},\ \Phi = \{\overline{5.6}, \overline{3.4}, \overline{1.2}\} = \pi.$$

La procédure de reconstitution de l'automate minimal d'après la partition π est la suivante.

Tableau 3-21

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_3	a_4	a_3	a_4	a_5	a_6
z_2	a_5	a_6	a_5	a_6	a_1	a_2

Tableau 3-22

λ: $A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_1	w_1	w_1	w_1	w_1
z_2	w_1	w_1	w_2	w_2	w_1	w_1

1. Choisir, dans chaque classe d'équivalence de la partition π, un élément; former avec ces éléments l'ensemble A' des états de l'automate minimal $S' = (A', Z, W, \delta', \lambda', a_1')$ équivalent à l'automate donné $S = (A, Z, W, \delta, \lambda, a_1)$. Dans notre exemple, nous choisirons $A' = \{a_1, a_4, a_5\}$.

2. Définir les fonctions des transitions $\delta' : A' \times Z \to A'$ et des sorties $\lambda' : A' \times Z \to W$ de S' sur l'ensemble $A' \times Z$. A cet effet, supprimer dans les tables des transitions et des sorties les colonnes qui correspondent aux états non compris dans A' et remplacer tous les états dans les colonnes restantes de la table des transitions par leurs équivalents de l'ensemble A'. L'automate minimal S_5' est présenté dans les tableaux 3-24, 3-25.

Tableau 3-23

Table triangulaire de l'automate S_5

	1	2	3	4	5
2	3.4 5.6				
3	××	××			
4	××	××	5.6		
5	3.5× ×	4.5× 1.6×	××	××	
6	3.6× 2.5×	4.6× ×	××	××	1.2

3. Comme a_1', choisir dans A' un état équivalent à a_1. Il est commode de prendre pour a_1' l'état a_1 lui-même, dans le cas où $a_1 \in A'$.

Tableau 3-24

$\delta' : A' \times Z \to A'$

	a_1	a_4	a_5
z_1	a_4	a_4	a_5
z_2	a_5	a_5	a_1

Tableau 3-25

$\lambda' : A' \times Z \to W$

	a_1	a_4	a_5
z_1	w_1	w_1	w_1
z_2	w_1	w_2	w_1

Considérons encore une autre méthode de recherche de la partition π de l'ensemble d'états d'un automate complètement spécifié en classes d'équivalence ; cette méthode a été proposée par D. D. Aufenkamp et F. E. Hohn [2].

Introduisons tout d'abord une relation d'équivalence moins forte : nous dirons que deux états a_m, a_s sont *k-équivalents* (notation $a_m \overset{k}{\equiv} a_s$) si l'on a $\omega(a_m, \xi_k) = \omega(a_s, \xi_k)$ pour tous les mots d'entrée ξ_k de longueur k. Deux états qui ne sont pas k-équivalents seront appelés *k-distinguables*.

L'algorithme de recherche de π est le suivant. On cherche par itérations successives les partitions $\pi_1, \pi_2, \ldots, \pi_k, \pi_{k+1}$ de l'ensemble A en classes d'états 1-, 2-, . . ., $k+1$-équivalents entre eux, jusqu'à ce que, à une $k+1$-ième itération, l'on constate que $\pi_{k+1} = \pi_k$. Les auteurs de l'algorithme montrent très simplement qu'à cette itération la partition $\pi_k = \pi$, c'est-à-dire que les états k-équivalents sont équivalents, et que le nombre d'itérations k conduisant à $\pi_k = \pi$ n'est jamais supérieur à $M - 1$, où M est la puissance de l'ensemble A.

A titre d'exemple, cherchons la partition π pour l'automate S_5 (tableaux 3-21, 3-22). D'après la table des sorties (tableau 3-22), nous obtenons immédiatement la partition π_1 en classes d'états 1-équivalents en regroupant en classes 1-équivalentes les colonnes identiques:

$$\pi_1 = \{B_1, B_2\} = \overline{1.2.5.6}, \overline{3.4.}$$

En effet, deux états sont 1-équivalents si, placé dans ces états et attaqué par un mot d'entrée de longueur 1 de son alphabet d'entrée, l'automate produit la même sortie, et ceci pour tout mot d'entrée, d'où l'on déduit que les colonnes de la table des sorties correspondant à ces états doivent être identiques.

Construisons le tableau de π_1 (tableau 3-26) en remplaçant les

Tableau 3-26

Partition π_1 des états de l'automate S_5

	B_1				B_2	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	B_2	B_2	B_1	B_1	B_2	B_2
z_2	B_1	B_1	B_1	B_1	B_1	B_1

Tableau 3-27

Partition π_2 des états de l'automate S_5

	C_1		C_2		C_3	
	a_1	a_2	a_5	a_6	a_3	a_4
z_1	C_3	C_3	C_2	C_2	C_3	C_3
z_2	C_2	C_2	C_1	C_1	C_2	C_2

états du tableau 3-21 par les classes de 1-équivalence correspondantes. Il est évident que deux états 1-équivalents a_m, a_s sont 2-équivalents s'ils sont 1-équivalents pour tout signal d'entrée comme précédemment. D'après le tableau 3-26, nous obtenons la partition π_2 en classes d'états 2-équivalents (tableau 3-27):

$$\pi_2 = \{C_1, C_2, C_3\} = \overline{1.2}, \overline{5.6}, \overline{3.4}.$$

La partition π_3 se construit de la même façon (tableau 3-27):

$$\pi_3 = \{D_1, D_2, D_3\} = \overline{1.2}, \overline{5.6}, \overline{3.4};$$

elle se confond avec π_2. La procédure est terminée. La partition $\pi_3 = \pi_2 = \pi$ est la partition de l'ensemble des états de l'automate de Mealy S_5 en classes d'équivalence.

L'algorithme de reconstitution d'un automate minimal équivalent à l'automate donné d'après sa partition π a été décrit plus haut.

Pour la minimisation des automates de Moore complètement spécifiés, on introduit la notion de 0-*équivalence* des états et la partition de l'ensemble d'états en classes 0-équivalentes: les états de l'automate de Moore repérés par le même signal sont regroupés dans une telle classe. Deux états 0-équivalents sont 1-équivalents s'ils sont 0-équivalents pour n'importe quel signal d'entrée. Toutes les autres classes d'équivalence des états de l'automate de Moore sont cherchées en analogie parfaite avec le cas de l'automate de Mealy examiné ci-dessus.

CHAPITRE 4

COMPOSITION ET DÉCOMPOSITION DES AUTOMATES

4-1. Connexion de deux automates

Dans les paragraphes qui suivront, nous définirons la notion de réseau d'automates et donnerons un énoncé des problèmes de composition et de décomposition d'automates. Le présent paragraphe contient la description des trois principaux types de connexion des automates: en parallèle, en série et avec boucle de retour (rétroaction).

Connexion en parallèle. Deux automates connectés en parallèle sont représentés sur la figure 4-1. Ici $S_1 = (A_1, Z, W_1, \delta_1, \lambda_1)$ et $S_2 = (A_2, Z, W_2, \delta_2, \lambda_2)$. L'alphabet d'entrée est commun aux deux automates. Les sorties de S_1 et de S_2 sont reliées à un traducteur de fonction φ (automate sans mémoire) qui réalise l'application $\varphi: W_1 \times W_2 \to W$.

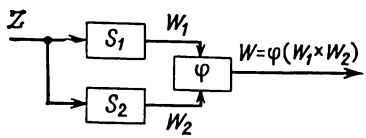


Fig. 4-1. Connexion en parallèle de deux automates

L'*automate résultant* de la connexion en parallèle de deux automates S_1, S_2 est l'automate $S = (A, Z, W, \delta, \lambda)$ dans lequel:

1. L'ensemble d'états $A = A_1 \times A_2$ est l'ensemble des couples dont le premier et le deuxième éléments sont respectivement les états des automates S_1 et S_2:

$$A = \{a_m = (a_{m_1}, a_{m_2}) \mid a_{m_1} \in A_1, a_{m_2} \in A_2\}.$$

2. L'alphabet d'entrée est l'alphabet d'entrée Z des automates S_1 et S_2.

3. L'alphabet de sortie est $W = \varphi(W_1 \times W_2)$, φ étant l'application à réaliser.

4. La fonction des transitions $\delta: A \times Z \to A$ se définit comme suit:

$$\delta(a_m, z_f) = (\delta_1(a_{m_1}, z_f), \delta_2(a_{m_2}, z_f)), \; z_f \in Z.$$

Une autre notation que nous utiliserons également est

$$\delta\,(A \times Z) = (\delta_1\,(A_1 \times Z),\ \delta_2\,(A_2 \times Z)).$$

5. La fonction des sorties $\lambda : A \times Z \to W$ se définit comme suit :

$$\lambda\,(a_m,\ z_f) = \varphi\,(\lambda_1\,(a_{m_1},\ z_f),\ \lambda_2\,(a_{m_2},\ z_f)),$$

ou encore

$$\lambda\,(A \times Z) = \varphi\,(\lambda_1\,(A_1 \times Z),\ \lambda_2\,(A_2 \times Z)).$$

Considérons un exemple de connexion en parallèle de deux automates $S_1 = (B,\ Z,\ U,\ \delta_1,\ \lambda_1)$ (tableaux 4-1 et 4-2) et $S_2 = (C,\ Z,\ V,\ \delta_2,\ \lambda_2)$ (tableaux 4-3 et 4-4). La fonction $\varphi : U \times V \to W$ est définie par le tableau 4-5.

Tableau 4-1

$\delta_1 : B \times Z \to B$

	b_1	b_2	b_3
z_1	b_1	b_1	b_2
z_2	b_3	b_3	b_2

Tableau 4-2

$\lambda_1 : B \times Z \to U$

	b_1	b_2	b_3
z_1	u_1	u_2	u_2
z_2	u_1	u_1	u_1

Tableau 4-3

$\delta_2 : C \times Z \to C$

	c_1	c_2
z_1	c_1	c_2
z_2	c_2	c_1

Tableau 4-4

$\lambda_2 : C \times Z \to V$

	c_1	c_2
z_1	v_1	v_2
z_2	v_2	v_1

Tableau 4-5

$\varphi : U \times V \to W$

	u_1	u_2
v_1	w_1	w_2
v_2	w_2	w_3

Tableau 4-6

$\delta : A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_1c_1	b_1c_2	b_1c_1	b_1c_2	b_2c_1	b_2c_1
z_2	b_3c_2	b_3c_1	b_3c_2	b_3c_1	b_2c_2	b_2c_1

L'automate résultant sera $S = (A, Z, W, \delta, \lambda)$, dans lequel:

1. $A = B \times C = \{(b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} = \{a_1, \ldots, a_6\}$.
2. $Z = \{z_1, z_2\}$.
3. $W = \{w_1, w_2, w_3\}$.
4. La fonction des transitions $\delta: A \times Z \to A$ est définie par le tableau 4-6. On a par exemple,

$$\delta(a_3, z_1) = \delta((b_2, c_1), z_1) = (\delta_1(b_2, z_1), \delta_2(c_1, z_1)) = (b_1, c_1) = a_1.$$

5. La fonction des sorties $\lambda: A \times Z \to W$ est définie par le tableau 4-7. On a, par exemple,

$$\lambda(a_3, z_1) = \lambda((b_2, c_1), z_1) = \varphi(\lambda_1(b_2, z_1), \lambda_2(c_1, z_1)) = \\ = \varphi(u_2, v_1) = w_2.$$

Tableau 4-7

$\lambda : A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_1	w_2	w_2	w_3	w_2	w_3
z_2	w_2	w_1	w_2	w_1	w_2	w_1

Connexion en série. Deux automates connectés en série sont représentés sur la figure 4-2. Ici $S_1 = (A_1, Z, W_1, \delta_1, \lambda_1)$ et $S_2 = (A_2, W_1, W, \delta_2, \lambda_2)$, c'est-à-dire que l'alphabet de sortie de l'automate S_1 est l'alphabet d'entrée de S_2.

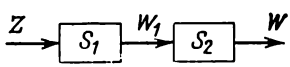


Fig. 4-2. Connexion en série de deux automates

L'automate résultant de la connexion en série de deux automates S_1, S_2 est $S = (A, Z, W, \delta, \lambda)$, dans lequel:

1. $A = A_1 \times A_2$, ou $A = \{a_m = (a_{m_1}, a_{m_2}) \mid a_{m_1} \in A_1, a_{m_2} \in A_2\}$.
2. $Z = Z$.
3. $W = W$.
4. La fonction $\delta: A \times Z \to A$ se définit comme suit:

$$\delta(a_m, z_f) = (\delta_1(a_{m_1}, z_f), \delta_2(a_{m_2}, \lambda_1(a_{m_1}, z_f))),$$

ou encore

$$\delta(A \times Z) = (\delta_1(A_1 \times Z), \delta_2(A_2 \times \lambda_1(A_1 \times Z))).$$

5. La fonction $\lambda: A \times Z \to W$ se définit comme suit:

$$\lambda (a_m, z_f) = \lambda_2 (a_{m_2}, \lambda_1 (a_{m_1}, z_f)),$$

ou encore

$$\lambda (A \times Z) = \lambda_2 (A_2 \times \lambda_1 (A_1 \times Z)).$$

Considérons un exemple d'interconnexion en série de deux automates $S_1 = (B, Z, U, \delta_1, \lambda_1)$ et $S_2 = (C, U, W, \delta_2, \lambda_2)$. L'automate S_1 est le même (tableaux 4-1, 4-2) que dans l'exemple de connexion en parallèle, tandis que l'automate S_2 est défini par les tableaux 4-8 et 4-9.

Tableau 4-8

$\delta_2 : C \times U \to C$

	c_1	c_2
u_1	c_1	c_2
u_2	c_2	c_1

Tableau 4-9

$\lambda_2 : C \times U \to W$

	c_1	c_2
u_1	w_1	w_2
u_2	w_2	w_1

L'automate résultant de cette connexion sera $S = (A, Z, W, \delta, \lambda)$ dans lequel:

1. $A = B \times C = \{(b_1, c_1), (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$.
2. $Z = \{z_1, z_2\}$.
3. $W = \{w_1, w_2\}$.
4. La fonction des transitions $\delta: A \times Z \to A$ est définie dans le tableau 4-10. Ici on a par exemple

$$\delta (a_3, z_1) = \delta ((b_2, c_1), z_1) = (\delta_1 (b_2, z_1), \delta_2 (c_1, \lambda_1 (b_2, z_1))) =$$
$$= (b_1, \delta_2 (c_1, u_2)) = (b_1, c_2) = a_2.$$

Tableau 4-10

$\delta : A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_1c_1	b_1c_2	b_1c_2	b_1c_1	b_2c_2	b_2c_1
z_2	b_3c_1	b_3c_2	b_3c_1	b_3c_2	b_2c_1	b_2c_2

5. La fonction des sorties $\lambda: A \times Z \to W$ est définie dans le tableau 4-11. On a par exemple $\lambda (a_3, z_1) = \lambda ((b_2, c_1), z_1) = \lambda_2 (c_1, \lambda_1 (b_2, z_1)) = \lambda_2 (c_1, u_2) = w_2$.

Tableau 4-11

$\lambda : A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_1	w_2	w_2	w_1	w_2	w_1
z_2	w_1	w_2	w_1	w_2	w_1	w_2

Connexion à boucle de retour. Deux automates connectés avec une boucle de retour sont représentés sur la figure 4-3. Ici $S_1 = (A_1, Z_1, W, \delta_1, \lambda_1)$ et $S_2 = (A_2, W, W_2, \delta_2, \lambda_2)$. Le traducteur de fonction (automate sans mémoire) réalise l'application $\gamma : Z \times W_2 \to Z_1$.

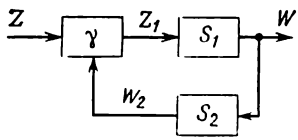


Fig. 4-3. Connexion à boucle de retour de deux automates

Nous allons montrer que dans le cas d'une connexion à boucle de retour, l'un des automates au moins — S_1 ou S_2 — est toujours un automate de Moore. Supposons le contraire: soient S_1, S_2 deux automates de Mealy. Alors

$$W_2 = \lambda_2 (A_2 \times W), \quad W = \lambda_1 (A_1 \times Z_1).$$

Le signal d'entrée Z_1 du premier automate serait

$$Z_1 = \gamma (Z \times W_2) = \gamma (Z \times \lambda_2 (A_2 \times W)) = \\ = \gamma (Z \times \lambda_2 (A_2 \times \lambda_1 (A_1 \times Z_1))).$$

On voit donc que Z_1 dépend, à un instant donné, de Z_1 au même instant, ce qui compromet la stabilité du réseau ainsi constitué. Par contre, si un des automates, par exemple S_1, est un automate de Moore, on a $W = \lambda_1 (A_1)$ et

$$Z_1 = \gamma (Z \times W_2) = \gamma (Z \times \lambda_2 (A_2 \times W)) = \\ = \gamma (Z \times \lambda_2 (A_2 \times \lambda_1 (A_1))).$$

Le réseau est stable.

Soit S_2 un modèle de Moore ($W_2 = \lambda_2 (A_2)$). L'automate résultant de la connexion à boucle de deux automates S_1, S_2 est $S = (A, Z, W, \delta, \lambda)$ dans lequel:

1. $A = A_1 \times A_2$, ou $A = \{a_m = (a_{m_1}, a_{m_2}) \mid a_{m_1} \in A_1, a_{m_2} \in A_2\}$.
2. $Z = Z$.
3. $W = W$.

4. La fonction $\delta: A \times Z \rightarrow A$ se définit comme suit:

$$\delta(a_m, z_f) = (\delta_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))), \ \delta_2(a_{m_2}, \lambda_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))))),$$

ou encore

$$\delta(A \times Z) = (\delta_1(A_1 \times \gamma(Z \times \lambda_2(A_2))),$$
$$\delta_2(A_2 \times \lambda_1(A_1 \times \gamma(Z \times \lambda_2(A_2))))).$$

5. La fonction $\lambda: A \times Z \rightarrow W$ se définit comme suit:

$$\lambda(a_m, z_f) = \lambda_1(a_{m_1}, \gamma(z_f, \lambda_2(a_{m_2}))),$$

ou encore

$$\lambda(A \times Z) = \lambda_1(A_1 \times \gamma(Z \times \lambda_2(A_2))).$$

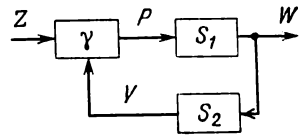


Fig. 4-4. Exemple de connexion de deux automates avec une boucle de retour

Considérons un exemple de connexion à boucle de retour (fig. 4-4) de deux automates $S_1 = (B, P, W, \delta_1, \lambda_1)$ (tableaux 4-12, 4-13) et $S_2 = (C, W, V, \delta_2, \lambda_2)$ (tableau 4-14). La fonction $\gamma: Z \times V \rightarrow P$ est définie dans le tableau 4-15.

Tableau 4-12

$\delta_1 : B \times P \rightarrow B$

	b_1	b_2	b_3
p_1	b_3	b_2	b_3
p_2	b_2	b_1	b_1

Tableau 4-13

$\lambda_1 : B \times P \rightarrow W$

	b_1	b_2	b_3
p_1	w_1	w_2	w_1
p_2	w_3	w_1	w_2

Tableau 4-14

$\delta_2 : C \times W \rightarrow C$,
$\lambda_2 : C \rightarrow V$

	v_1	v_2
	c_1	c_2
w_1	c_1	c_2
w_2	c_2	c_2
w_3	c_1	c_1

Tableau 4-15

$\gamma : Z \times V \rightarrow P$

	z_1	z_2	z_3
v_1	p_1	p_1	p_1
v_2	p_2	p_2	p_1

L'automate résultant de cette connexion est $S = (A, Z, W, \delta, \lambda)$ dans lequel:

1. $A = B \times C = \{(b_1, c_1,) (b_1, c_2), (b_2, c_1), (b_2, c_2), (b_3, c_1), (b_3, c_2)\} = \{a_1, a_2, a_3, a_4, a_5, a_6\}$.

2. $Z = \{z_1, z_2, z_3\}$.

3. $W = \{w_1, w_2, w_3\}$.

4. La fonction des transitions $\delta: A \times Z \rightarrow A$ est définie dans le tableau 4-16. On a par exemple $\delta(a_3, z_1) = \delta((b_2, c_1), z_1) = (\delta_1(b_2, \gamma(z_1, \lambda_2(c_1))), \delta_2(c_1, \lambda_1(b_2, \gamma(z_1, \lambda_2(c_1))))) = (\delta_1(b_2, \gamma(z_1, v_1)), \delta_2(c_1, \lambda_1(b_2, \gamma(z_1, v_1)))) = (\delta_1(b_2, p_1), \delta_2(c_1, \lambda_1(b_2, p_1))) = (b_2, \delta_2(c_1, w_2)) = (b_2, c_2) = a_4$.

Tableau 4-16

$\delta: A \times Z \rightarrow A$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	b_3c_1	b_2c_1	b_2c_2	b_1c_2	b_3c_1	b_1c_2
z_2	b_3c_1	b_2c_1	b_2c_2	b_1c_2	b_3c_1	b_1c_2
z_3	b_3c_1	b_3c_2	b_2c_2	b_2c_2	b_3c_1	b_3c_2

Tableau 4-17

$\lambda: A \times Z \rightarrow W$

	a_1	a_2	a_3	a_4	a_5	a_6
	b_1c_1	b_1c_2	b_2c_1	b_2c_2	b_3c_1	b_3c_2
z_1	w_1	w_3	w_2	w_1	w_1	w_2
z_2	w_1	w_3	w_2	w_1	w_1	w_2
z_3	w_1	w_1	w_2	w_2	w_1	w_1

5. La fonction des sorties $\lambda: A \times Z \rightarrow W$ est définie dans le tableau 4-17. On a par exemple $\lambda(a_3, z_1) = \lambda((b_2, c_1), z_1) = \lambda_1(b_2, \gamma(z_1, \lambda_2(c_1))) = \lambda_1(b_2, \gamma(z_1, v_1)) = \lambda_1(b_2, p_1) = w_2$.

4-2. Connexion à fonction de sortie

Considérons trois automates $S_1 = (B, Z, X, \delta_1, \lambda_1)$, $S_2 = (C, X, Y, \delta_2, \lambda_2)$ et $S_3 = (D, Y, W, \delta_3, \lambda_3)$ connectés en série (fig. 4-5). Plaçons-nous dans le cas général où, à un instant initial i_1, les automates S_1, S_2 et S_3 se trouvent respectivement dans les états

b_{i_1}, c_{i_1} et d_{i_1} ; l'entrée de S_1 est attaquée par un mot d'entrée $\xi_z = z_{i_1} z_{i_2} \ldots z_{i_p}$. Pour connaître le mot de sortie ξ_w, il est nécessaire de définir la réponse de chaque automate :

Mot d'entrée ξ_z	z_{i_1}	z_{i_2}	. . .
Séquence d'états internes de S_1	b_{i_1}	$b_{i_2} = \delta_1 (b_{i_1}, z_{i_1})$	. . .
Mot de sortie de S_1 $\xi_x = \omega_1 (b_{i_1}, \xi_z)$	$x_{i_1} = \lambda_1 (b_{i_1}, z_{i_1})$	$x_{i_2} = \lambda_1 (b_{i_2}, z_{i_2})$	. . .
Séquence d'états internes de S_2	c_{i_1}	$c_{i_2} = \delta_2 (c_{i_1}, x_{i_1})$	. . .
Mot de sortie de S_2 $\xi_y = \omega_2 (c_{i_1}, \xi_x)$	$y_{i_1} = \lambda_2 (c_{i_1}, x_{i_1})$	$y_{i_2} = \lambda_2 (c_{i_2}, x_{i_2})$	. . .
Séquence d'états internes de S_3	d_{i_1}	$d_{i_2} = \delta_3 (d_{i_1}, y_{i_1})$	. . .
Mot de sortie de S_3 $\xi_w = \omega_3 (d_{i_1}, \xi_y)$	$w_{i_1} = \lambda_3 (d_{i_1}, y_{i_1})$	$w_{i_2} = \lambda_3 (d_{i_2}, y_{i_2})$	. . .

Il est clair que S_2 ne peut produire le signal de sortie $y_{i_1} = \lambda_2 (c_{i_1}, x_{i_1})$ et passer à l'état suivant $c_{i_2} = \delta_2 (c_{i_1}, x_{i_1})$ qu'après que l'automate S_1 ait produit son signal de sortie $x_{i_1} = \lambda_1 (b_{i_1}, z_{i_1})$. Tout pareillement, S_3 ne peut produire $w_{i_1} = \lambda_3 (d_{i_1}, y_{i_1})$ et passer à l'état suivant avant d'avoir reçu le signal de sortie y_{i_1} produit par S_2. Ainsi donc, le signal de sortie w_{i_1}, bien que se rapportant à l'instant i_1, apparaîtra à la sortie du système au bout de $3T$, où T est le temps mis par chacun des automates pour élaborer son signal de sortie. Dans le cas de connexion en série de n automates, ce temps de retard atteindra nT, ce qui peut entraver le fonctionnement du système. En outre, un tel mode de connexion risque de perturber gravement la synchronisation des automates, surtout lorsqu'on adopte un couplage mixte série — parallèle ou qu'on introduit des liaisons en retour.

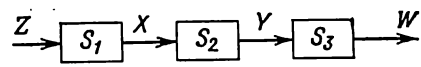


Fig. 4-5. Connexion en série de trois automates

Au lieu de la structure représentée sur la figure 4-5, nous examinerons donc une connexion des automates dite *à fonction de sortie* (fig. 4-6). Les automates précédents définis par un quintuple $S_i = (A_i, Z_i, W_i, \delta_i, \lambda_i)$ sont remplacés par des *semi-automates* $S_i = (A_i, Z_i, \delta_i)$. On entend par-là un automate de Moore à sorties *complètes*, c'est-à-dire qu'à chaque état interne de l'automate correspond son propre signal de sortie, distinct de ceux des autres états internes. Il est évident que dans un tel automate il y a autant de

signaux de sortie que d'états internes différents. Compte tenu de ce qui précède, on comprend que dans un automate de Moore à sorties complètes, il est possible d'identifier les états aux signaux de sortie. On peut donc utiliser, dans un semi-automate, le même alphabet pour les états internes et pour les signaux de sortie. Il est clair que la fonction des sorties devient inutile en pareils cas: en définissant

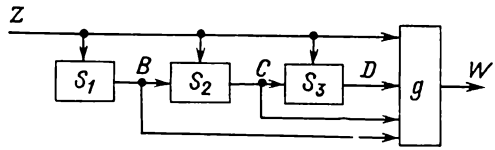


Fig. 4-6. Connexion en série à fonction de sortie

un semi-automate, on utilise simplement sa table des transitions. Les semi-automates sont quelquefois appelés automates sans sortie.

Nous admettons donc que le circuit de la figure 4-6 est constitué par les semi-automates $S_1 = (B, Z, \delta_1)$, $S_2 = (C, B \times Z, \delta_2)$ et $S_3 = (D, C \times Z, \delta_3)$. La fonction de sortie réalise l'application $g: B \times C \times D \times Z \rightarrow W$. Si, à un instant initial i_1, les automates S_1, S_2, S_3 se trouvent dans les états b_{i_1}, c_{i_1} et d_{i_1} respectivement, l'arrivée d'un signal d'entrée z_{i_1} déclenche dans tous les trois automates la transition à l'état suivant:

$$b_{i_2} = \delta_1(b_{i_1}, z_{i_1}), \quad c_{i_2} = \delta_2(c_{i_1}, (b_{i_1}, z_{i_1})), \quad d_{i_2} = \delta_3(d_{i_1}, (c_{i_1}, z_{i_1})),$$

car les entrées de S_2 et S_3 sont, à l'instant i_1, les états actuels b_{i_1} et c_{i_1} de S_1 et S_2 respectivement, ainsi que le signal d'entrée z_{i_1}. Le signal de sortie $w_{i_1} = g(b_{i_1}, c_{i_1}, d_{i_1}, z_{i_1})$ est aussi fonction des états actuels à l'instant i_1 et du signal d'entrée z_{i_1} au même instant. Ainsi donc, l'augmentation du nombre d'automates montés en série n'augmente nullement le retard à la sortie.

Remarquons que c'est la fonction de sortie g qui détermine si l'automate résultant est un automate de Mealy ou de Moore: dans le dernier cas g ne dépend pas de Z.

4-3. Réseau d'automates

Un modèle utilisé en théorie algébrique des automates [28] pour décrire le fonctionnement en commun d'un groupe d'automates est le réseau d'automates. La notion de réseau d'automates telle qu'on la définit dans ce paragraphe est assez voisine de celle adoptée dans [28]. Or, les auteurs du travail cité ne définissent pas la notion de structure de réseau; par contre, nous estimons que la structure de réseau, que nous définissons comme une collection de fonctions, est fort utile dans l'approche des problèmes de composition et de décom-

position de réseaux ainsi que dans l'étude de leurs propriétés structurelles.

La plupart des définitions de ce paragraphe sont tirées de l'article [35] dû à A. Keevallik *).

Le réseau d'automates est un sextuple

$$N = (Z,\ \{S_i\},\ W,\ \{f_i\},\ \{\psi_i\},\ g)$$

dans lequel :

1. Z est l'alphabet d'entrée.

2. $\{S_i = (A_i,\ Z_i,\ \delta_i)\}$, $1 \leqslant i \leqslant n$, est un ensemble d'*automates composants* (AC) *du réseau*. Chaque automate composant S_i est un sous-automate (fig. 4-7) avec son alphabet d'entrée Z_i,

$$Z_i = \begin{cases} Z_i' \times Z_i'' & \text{pour } Z_i' \neq \varnothing, \\ Z_i'' & \text{pour } Z_i' = \varnothing, \end{cases}$$

où Z_i' est son *alphabet d'entrée interne* et Z_i'' son *alphabet d'entrée externe*.

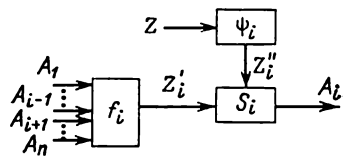


Fig. 4-7. Automate composant d'un réseau

La fonction des transitions de S_i est $\delta_i : A_i \times Z_i \to A_i$.

3. W est l'alphabet de sortie du réseau.

4. $\{f_i : (\underset{j}{\times} A_j \to Z_i'\}$, $1 \leqslant i,\ j \leqslant n$ est l'ensemble des *fonctions de connexion* des automates composants du réseau **).

5. $\{\psi_i : Z \to Z_i''\}$, $1 \leqslant i \leqslant n$, est l'ensemble des *fonctions d'entrée*.

6. $g : (\underset{i}{\times} A_i) \times Z \to W$ est la *fonction de sortie du réseau*.

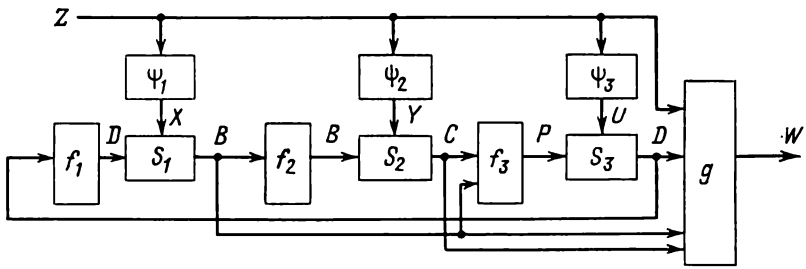


Fig. 4-8. Réseau composé de n automates

Les ensembles $\{S_i\}$ et $\{f_i\}$ s'appellent respectivement la *base* et la *structure* du réseau.

*) A. Mélikhov (voir par exemple [44]) propose une approche légèrement différente du problème de décomposition.

**) $\underset{j}{\times} A_j$ est la notation abrégée de $A_1 \times A_2 \times \ldots \times A_n$.

Un réseau ainsi défini peut être considéré comme un modèle général susceptible de décrire le travail en commun de plusieurs automates réunis dans un système, car il ne comporte aucune restriction quant au choix des composants ou de la structure. Un réseau d'automates à n composants est représenté sur la figure 4-8.

Tout réseau N admet un automate fonctionnellement équivalent S_N que nous appellerons *automate résultant* du réseau N.

Un réseau $N = (Z, \{S_i\}, W, \{f_i\}, \{\psi_i\}, g)$ admet un automate résultant

$$S_N = (A_N, Z_N, W_N, \delta_N, \lambda_N)$$

dans lequel:

1. $A_N = \underset{i}{\times} A_i, \ i = 1, \ldots, n.$
2. $Z_N = Z.$
3. $W_N = W.$
4. La fonction des transitions $\delta_N: A_N \times Z_N \to A_N$ se définit comme suit:

$$\delta_N(a_m, z_f) = \delta_N((a_{m_1}, \ldots, a_{m_i}, \ldots, a_{m_n}), z_f) =$$
$$= \underset{i}{\times} \delta_i(a_{m_i}, (f_i(a_{m_1}, \ldots, a_{m_n}), \psi_i(z_f)))^{*)}.$$

Ici $a_m \in A_N$, $z_f \in Z_N$, $a_{m_i} \in A_i$, $a_m = (a_{m_1}, \ldots, a_{m_n})$.

5. La fonction des sorties $\lambda_N: A_N \times Z_N \to W_N$ (pour le modèle de Mealy) se définit comme suit:

$$\lambda_N(a_m, z_f) = g((a_{m_1}, \ldots, a_{m_n}), z_f).$$

Dans le modèle de Moore on a

$$\lambda_N: A_N \to W_N, \quad \lambda_N(a_m) = g(a_{m_1}, \ldots, a_{m_n}).$$

Prenons un exemple. Soit un réseau $N = (Z, \{S_1, S_2, S_3\}, W, \{f_1, f_2, f_3\}, \{\psi_1, \psi_2, \psi_3\}, g)$ (fig. 4-9) dans lequel $Z = \{z_1, z_2, z_3, z_4\}$, $W = \{w_1, w_2\}$.

---

*) $\underset{i}{\times}\delta_i(a_{m_i}, (f_i(a_{m_1}, \ldots, a_{m_n}), \psi_i(z_f)))$ est la notation abrégée d'un vecteur à n composantes dont la i-ième composante est égale à $\delta_i(a_{m_i}, (f_i(a_{m_1}, \ldots, a_{m_n}), \psi_i(z_f)))$.

L'automate composant $S_1 = (B, Z_1, \delta_1)$. Son alphabet d'entrée est $Z_1 = X$, car S_1 ne reçoit pas en entrée de signaux issus d'autres automates composants; $X = \{x_1, x_2, x_3\}$. La fonction $f_1 = \varnothing$;

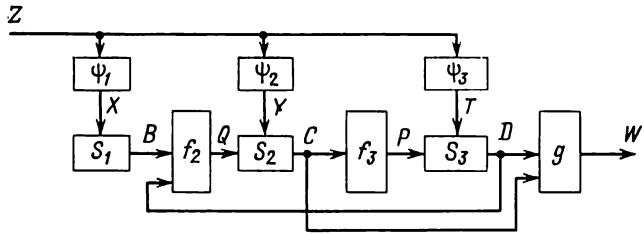


Fig. 4-9. Exemple de réseau composé de trois automates

la fonction $\psi_1 : Z \to X$ est définie dans le tableau 4-18. La fonction des transitions $\delta_1 : B \times X \to B$ est définie dans le tableau 4-19.

Tableau 4-18

$\psi_1 : Z \to X$

Z	X
z_1	x_1
z_2	x_1
z_3	x_2
z_4	x_3

Tableau 4-19

$\delta_1 : B \times X \to B$

	b_1	b_2	b_3
x_1	b_1	b_1	b_3
x_2	b_3	b_2	b_3
x_3	b_3	b_2	b_1

Tableau 4-20

$\psi_2 : Z \to Y$

Z	Y
z_1	y_1
z_2	y_2
z_3	y_1
z_4	y_2

Tableau 4-21

$f_2 : B \times D \to Q$

$B \times D$	Q
b_1d_1	q_1
b_1d_2	q_2
b_2d_1	q_1
b_2d_2	q_1
b_3d_1	q_1
b_3d_2	q_1

Tableau 4-22

$\delta_2 : C \times (Q \times Y) \to C$

	c_1	c_2
q_1y_1	c_1	c_1
q_1y_2	c_2	c_1
q_2y_1	c_1	c_2
q_2y_2	c_2	c_2

L'automate composant $S_2 = (C, Z_2, \delta_2)$. Son alphabet d'entrée est $Z_2 = Q \times Y = \{(q_1, y_1), (q_1, y_2), (q_2, y_1), (q_2, y_2)\}$; $Q = \{q_1, q_2\}$; $Y = \{y_1, y_2\}$.

Tableau 4-23

$\psi_3 : Z \to T$

Z	T
z_1	t_1
z_2	t_1
z_3	t_1
z_4	t_2

Tableau 4-24

$f_3 : C \to P$

C	P
c_1	p_1
c_2	p_2

Tableau 4-25

$\delta_3 : D \times (P \times T) \to D$

	d_1	d_2
p_1t_1	d_2	d_1
p_1t_2	d_1	d_2
p_2t_1	d_2	d_2
p_2t_2	d_1	d_1

Tableau 4-26

$g : C \times D \to W$

$C \times D$	W
c_1d_1	w_1
c_1d_2	w_2
c_2d_1	w_1
c_2d_2	w_1

Les fonctions $\psi_2 : Z \to Y$, $f_2 : B \times D \to Q$ et $\delta_2 : C \times (Q \times Y) \to C$ sont définies dans les tableaux 4-20, 4-21 et 4-22 respectivement.

L'automate composant $S_3 = (D, Z_3, \delta_3)$. Son alphabet d'entrée est

$$Z_3 = P \times T = \{(p_1, t_1), (p_1, t_2), (p_2, t_1), (p_2, t_2)\};$$
$$P = \{p_1, p_2\}, \quad T = \{t_1, t_2\}.$$

Les fonctions $\psi_3 : Z \to T$, $f_3 : C \to P$ et $\delta_3 : D \times (P \times T) \to D$ sont définies dans les tableaux 4-23, 4-24 et 4-25 respectivement.

La fonction de sortie du réseau $g : C \times D \to W$ est définie dans le tableau 4-26.

Tableau 4-27

Table des codes d'états de S_N

A_N	$B \times C \times D$
a_1	$b_1c_1d_1$
a_2	$b_1c_1d_2$
a_3	$b_1c_2d_1$
a_4	$b_1c_2d_2$
a_5	$b_2c_1d_1$
a_6	$b_2c_1d_2$
a_7	$b_2c_2d_1$
a_8	$b_2c_2d_2$
a_9	$b_3c_1d_1$
a_{10}	$b_3c_1d_2$
a_{11}	$b_3c_2d_1$
a_{12}	$b_3c_2d_2$

Tous les paramètres énumérés sont des données du problème, puisque le réseau N est donné. On demande de composer son automate résultant $S_N = (A_N, Z_N, W_N, \delta_N, \lambda_N)$ dans lequel:

1. $A_N = B \times C \times D = \{(b_1, c_1, d_1), \ldots, (b_3, c_2, d_2)\} = \{a_1, \ldots, a_{12}\}$. Si le nombre d'états est grand, il est bon de définir A_N à l'aide d'une table des codes d'états (tableau 4-27) dans laquelle à chaque état de A_N on fait correspondre un vecteur ayant pour composantes les états des automates composants du réseau.
2. $Z_N = Z = \{z_1, z_2, z_3, z_4\}$.
3. $W_N = W = \{w_1, w_2\}$.
4. La fonction des transitions $\delta_N : A_N \times Z_N \to A_N$ est définie dans le tableau 4-28 (table des transitions marquée, car S_N est un automate de Moore: la fonction de sortie g est indépendante de Z). On a par exemple $\delta_N(a_3, z_1) = \delta_N((b_1, c_2, d_1), z_1) = (\delta_1(b_1, \psi_1(z_1)), \delta_2(c_2, (f_2(b_1, d_1), \psi_2(z_1))), \delta_3(d_1, (f_3(c_2), \psi_3(z_1)))) = (\delta_1(b_1, x_1), \delta_2(c_2, (q_1, y_1)), \delta_3(d_1, (p_2, t_1))) = (b_1, c_1, d_2) = a_2$.

5. La fonction des sorties λ_N : $: A_N \to W_N$ est définie dans le même tableau 4-28. Par exemple, $\lambda_N(a_3) = \lambda_N(b_1, c_2, d_1) = g(c_2, d_1) = w_1$.

Les notions de réseau d'automates et d'automate résultant permettent de formuler les problèmes de composition et de décomposition des automates. Par *composition d'automates* on entend la recherche de l'automate résultant S_N pour un réseau donné N.

Quant au problème de décomposition, il ne peut être formulé sans introduire quelques définitions préliminaires.

Nous disons qu'un automate $S' = (A', Z', W', \delta', \lambda')$ est un sous-automate d'un automate $S = (A, Z, W, \delta, \lambda)$ si et seulement si pour $A' \subseteq A$, $Z' \subseteq Z$, $W' \subseteq W$ et pour tout $a_m \in A'$ et tout $z_f \in Z'$ on a

$$\delta'(a_m, z_f) = \delta(a_m, z_f),$$
$$\lambda'(a_m, z_f) = \lambda(a_m, z_f).$$

Autrement dit, les deux automates se comportent de la façon identique dans tout le domaine de définition de S'. Ainsi donc, l'automate S « fait autant sinon davantage que S' ».

On donne à titre d'exemple deux automates de Mealy, S' et S,

Tableau 4-29

$\delta': A' \times Z' \to A'$

	a_1	a_2	a_3
z_1	a_2	a_1	a_3
z_2	a_3	a_2	a_1

Tableau 4-30

$\lambda': A' \times Z' \to W'$

	a_1	a_2	a_3
z_1	w_1	w_3	w_3
z_2	w_2	w_1	w_2

Tableau 4-28

$\delta_N: A_N \times Z_N \to A_N$, $\lambda_N: A_N \to W_N$

W	w_1	w_2	w_1	w_1	w_2	w_1	w_1	w_1	w_2	w_1	w_1
A	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
	$b_1c_1d_1$	$b_1c_1d_2$	$b_1c_2d_1$	$b_1c_2d_2$	$b_2c_1d_1$	$b_2c_1d_2$	$b_2c_2d_1$	$b_2c_2d_2$	$b_3c_1d_1$	$b_3c_1d_2$	$b_3c_2d_1$	$b_3c_2d_2$
z_1	$b_1c_1d_2$	$b_1c_1d_1$	$b_1c_1d_2$	$b_1c_2d_2$	$b_1c_1d_2$	$b_1c_1d_1$	$b_1c_1d_2$	$b_1c_1d_2$	$b_3c_1d_2$	$b_3c_1d_1$	$b_3c_1d_2$	$b_3c_1d_2$
z_2	$b_1c_2d_2$	$b_1c_2d_1$	$b_1c_1d_2$	$b_1c_2d_2$	$b_1c_2d_2$	$b_1c_2d_1$	$b_1c_2d_2$	$b_1c_2d_2$	$b_3c_2d_2$	$b_3c_2d_1$	$b_3c_2d_2$	$b_3c_2d_2$
z_3	$b_3c_1d_2$	$b_3c_1d_1$	$b_3c_1d_2$	$b_3c_2d_2$	$b_2c_1d_2$	$b_2c_1d_1$	$b_2c_1d_2$	$b_2c_1d_2$	$b_3c_1d_2$	$b_3c_1d_1$	$b_3c_1d_2$	$b_3c_1d_2$
z_4	$b_3c_2d_1$	$b_3c_2d_2$	$b_3c_1d_1$	$b_3c_2d_1$	$b_2c_2d_1$	$b_2c_2d_2$	$b_2c_1d_1$	$b_2c_1d_1$	$b_1c_2d_1$	$b_1c_2d_2$	$b_1c_1d_1$	$b_1c_1d_1$

S' étant un sous-automate de S, dans les tableaux 4-29, 4-30 et 4-31, 4-32 respectivement.

Tableau 4-31

$\delta : A \times Z \to A$

	a_1	a_2	a_3	a_4
z_1	a_2	a_1	a_3	a_4
z_2	a_3	a_2	a_1	a_1
z_3	a_4	a_2	a_4	a_3

Tableau 4-32

$\lambda : A \times Z \to W$

	a_1	a_2	a_3	a_4
z_1	w_1	w_3	w_3	w_4
z_2	w_2	w_1	w_2	w_2
z_3	w_1	w_4	w_1	w_1

On dit que deux automates S, S' sont *isomorphes* s'il existe trois applications biunivoques

$$\varphi_1 : A \to A', \quad \varphi_2 : Z \to Z', \quad \varphi_3 : W \to W'$$

telles que

$$\varphi_1 (\delta (a_m, z_f)) = \delta' (\varphi_1 (a_m), \varphi_2 (z_f))$$

et

$$\varphi_3 (\lambda (a_m, z_f)) = \lambda' (\varphi_1 (a_m), \varphi_2 (z_f))$$

pour tout $a_m \in A$ et pour tout $z_f \in Z$.

Le triplet d'applications φ_1, φ_2, φ_3 s'appelle *isomorphisme* des automates S et S'. La notion d'isomorphisme se définit brièvement en disant que l'image de la fonction est égale à la fonction des images. En d'autres mots, deux automates isomorphes sont identiques, aux notations près des états internes, des signaux d'entrée et des signaux de sortie.

Deux automates isomorphes $S = (A, Z, W, \delta, \lambda)$ et $S' = (B, U, V, \delta', \lambda')$ sont définis dans les tableaux 4-33, 4-34 et 4-35, 4-36 respectivement.

Ici l'isomorphisme se résume par le triplet φ_1, φ_2, φ_3:

$$\begin{array}{lll} \varphi_1 : a_1 \to b_2 & \varphi_2 : z_1 \to u_2 & \varphi_3 : w_1 \to v_1 \\ \quad a_2 \to b_1 & \quad z_2 \to u_1 & \quad w_2 \to v_3 \\ \quad a_3 \to b_3 & & \quad w_3 \to v_2. \end{array}$$

Vérifions l'isomorphisme pour le couple (a_2, z_1) par exemple: $\varphi_1 (\delta (a_2, z_1)) = \varphi_1 (a_1) = b_2$, $\delta' (\varphi_1 (a_2), \varphi_2 (z_1)) = \delta' (b_1, u_2) = b_2$, $\varphi_3 (\lambda (a_2, z_1)) = \varphi_3 (w_3) = v_2$, $\lambda' (\varphi_1 (a_2), \varphi_2 (z_1)) = \lambda' (b_1, u_2) = v_2$.

Tableau 4-33

$\delta : A \times Z \to A$

	a_1	a_2	a_3
z_1	a_2	a_1	a_3
z_2	a_3	a_2	a_1

Tableau 4-34

$\lambda : A \times Z \to W$

	a_1	a_2	a_3
z_1	w_1	w_3	w_3
z_2	w_2	w_1	w_2

Tableau 4-35

$\delta' : B \times U \to B$

	b_1	b_2	b_3
u_1	b_1	b_3	b_2
u_2	b_2	b_1	b_3

Tableau 4-36

$\lambda' : B \times U \to V$

	b_1	b_2	b_3
u_1	v_1	v_3	v_3
u_2	v_2	v_1	v_2

Nous disons que S est une *réalisation* de S' (notation $S = R\ (S')$) si l'automate S admet un sous-automate isomorphe à S'. Ainsi donc, si S réalise S', son comportement doit se laisser décrire par les mêmes fonctions, aux notations près, que celui de S' partout où ce dernier est défini, car S admet nécessairement un sous-automate S'' isomorphe à S'.

Le *problème de décomposition* d'un automate S peut maintenant être formulé ainsi : on demande de construire un réseau N tel que son automate résultant S_N soit une réalisation de l'automate S donné :

$$S_N = R\ (S).$$

4-4. Partitions sur l'ensemble d'états de l'automate résultant d'un réseau

Rappelons que la *partition d'un ensemble* A est un ensemble $\pi = \{B_1, \ldots, B_m\}$ dont les éléments sont des parties de A ($B_i \subseteq \subseteq A$, $i = 1, \ldots, m$), telles que :

pour deux ensembles quelconques B_i et B_j ($i \neq j$) on a $B_i \cap \cap B_j = \varnothing$;

$$\bigcup_{i=1}^{m} B_i = A;$$

$$B_i \neq \varnothing,\ i = 1, \ldots, m.$$

Les B_i ($i = 1, \ldots, m$) sont les sous-ensembles (ensembles) de la partition π.

Soit par exemple $A = \{1, 2, 3, 4, 5, 6, 7\}$. La partition $\pi = \{\{1, 2, 4\}, \{3\}, \{5, 6, 7\}\}$ sera notée comme suit: $\pi = \overline{1.2.4}, \overline{3}, \overline{5.6.7}$. Autres exemples de partitions du même ensemble: $\overline{1}, \overline{2.3.4.5.6}, \overline{7}$; $\overline{1.2.3.4.5.6.7}$; $\overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}, \overline{6}, \overline{7}$.

Les deux dernières partitions sont un peu particulières. La partition constituée d'un ensemble unique s'appelle *partition unité* et se note π (1). La partition dont les ensembles sont des singlets s'appelle *partition zéro* et se note π (0).

Soient π_i, π_j deux partitions d'un ensemble A. Nous dirons que $\pi_i \leqslant \pi_j$ si et seulement si chaque ensemble de π_i est contenu dans un ensemble de π_j. Il est clair qu'une relation ainsi introduite est celle d'ordre partiel, car elle est

1) réflexive ($\pi_i \leqslant \pi_j$);
2) transitive (si $\pi_i \leqslant \pi_j$ et $\pi_j \leqslant \pi_p$, on a $\pi_i \leqslant \pi_p$);
3) antisymétrique (si $\pi_i \leqslant \pi_j$ et $\pi_j \leqslant \pi_i$, on a $\pi_i = \pi_j$).

Soient par exemple $\pi_1 = \overline{1.2.3.4}, \overline{5.6.7}$, $\pi_2 = \overline{1.2}, \overline{3.4}, \overline{5.6.7}$ et $\pi_3 = \overline{1.2.5.6}, \overline{3.4.7}$ trois partitions de l'ensemble $A = \{1, 2, 3, 4, 5, 6, 7\}$. Il est évident que $\pi_2 \leqslant \pi_1$ et que π_1, π_3 ne sont pas comparables ($\pi_1 \nleqslant \pi_3$ et $\pi_3 \nleqslant \pi_1$). Les partitions π_2, π_3 ne le sont non plus.

Il est clair que $\pi_i \leqslant \pi$ (1) et que π (0) $\leqslant \pi_i$, où π_i est une partition quelconque d'un ensemble et π (0), π (1) sont respectivement la partition zéro et la partition unité du même ensemble.

Introduisons, sur l'ensemble des partitions d'un ensemble donné, les opérations de produit et de somme de partitions. Soient π_i, π_j deux partitions d'un ensemble A. Par *produit* $\pi_i\pi_j$ *des partitions* π_i et π_j on entend une partition de A constituée par la totalité des intersections non vides de chaque ensemble de π_i avec chaque ensemble de π_j.

EXEMPLE. $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

$$\pi_1 = \overline{1.2.3.4}, \overline{5.6.7.8}; \ \pi_2 = \overline{1.3.5}, \overline{2.4.6.7.8};$$

$$\overline{1.2.3.4} \cap \overline{1.3.5} = \overline{1.3}; \ \overline{1.2.3.4} \cap \overline{2.4.6.7.8} = \overline{2.4};$$

$$\overline{5.6.7.8} \cap \overline{1.3.5} = \overline{5}; \ \overline{5.6.7.8} \cap \overline{2.4.6.7.8} = \overline{6.7.8};$$

$$\pi_3 = \pi_1\pi_2 = \overline{1.3}, \overline{2.4}, \overline{5}, \overline{6.7.8}.$$

Nous dirons que deux ensembles A et B sont *liés* si $A \cap B \neq \varnothing$. Nous disons que deux ensembles A et B sont *enchaînés* s'il existe une suite $A_1, A_2, \ldots, A_n$ telle que $A = A_1$, $B = A_n$ et que A_i est lié avec A_{i+1} pour tout $i = 1, \ldots, n - 1$ (c'est-à-dire $A_i \cap A_{i+1} \neq \varnothing$).

Définissons la *somme des partitions* $\pi = \pi_i + \pi_j$. Soit B un ensemble de π_i. Alors l'ensemble de π contenant B est la réunion de tous les ensembles de π_i et de π_j enchaînés à B.

EXEMPLE. $A = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

$$\pi_1 = \overline{1.5},\ \overline{2.6},\ \overline{3.8.9},\ \overline{4.7};\ \pi_2 = \overline{1.4},\ \overline{2.3.9},\ \overline{5.7},\ \overline{6.8};$$

$$\overline{1.5} \cap \overline{1.4} \neq \varnothing;$$

$$\overline{1.4} \cap \overline{4.7} \neq \varnothing;$$

$$\overline{4.7} \cap \overline{5.7} \neq \varnothing.$$

Alors $\overline{1.4.5.7}$ est un ensemble de $\pi_3 = \pi_1 + \pi_2$. De même, $\overline{2.3.6.8.9}$ est un deuxième ensemble de π_3:

$$\pi_1 + \pi_2 = \pi_3 = \overline{1.4.5.7},\ \overline{2.3.6.8.9}.$$

La somme des partitions $\pi_3 = \pi_1 + \pi_2$ se laisse parfaitement illustrer à l'aide d'un graphe. Traçons un graphe en assimilant ses sommets aux éléments de l'ensemble A. On a défini sur A deux partitions π_1 et π_2. Nous relions deux sommets p et q $(p, q \in A)$ par une arête si p et q font partie d'un même ensemble de π_1 ou de π_2: la partition π_3 contient alors autant d'ensembles qu'il y a de composantes connexes sur le graphe construit. Les sommets appartenant à une même composante forment un ensemble de la partition π_3. Le graphe de la figure 4-10 illustre le dernier exemple.

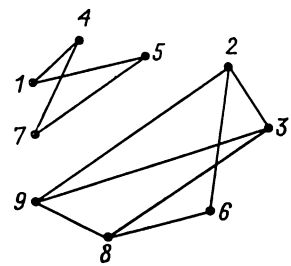


Fig. 4-10. Graphe illustrant la somme des partitions $\pi_1 = \overline{1.5},\ \overline{2.6},\ \overline{3.8.9},\ \overline{4.7}$ et $\pi_2 = \overline{1.4},\ \overline{2.3.9},\ \overline{5.7},\ \overline{6.8}$

Introduisons quelques notations: $a \equiv b\ (\pi)$ signifie que a et b sont contenus dans un même ensemble de la partition π; $B_\pi\ (a)$ désigne l'ensemble de la partition π qui contient a.

Passons aux partitions de l'ensemble d'états A_N de l'automate résultant $S_N = (A_N, Z_N, W_N, \delta_N, \lambda_N)$ d'un réseau N composé de n automates $S_1, \ldots, S_n$. Montrons que le comportement de chaque automate composant peut être défini en termes de partitions de A_N.

Soient $a_m = (a_{m_1}, \ldots, a_{mn})$, $a_p = (a_{p_1}, \ldots, a_{pn})$; $a_m, a_p \in A_N$; $a_{m_j}, a_{p_j} \in A_j$; A_j est l'ensemble d'états d'un automate composant S_j, $j = 1, \ldots, n$. Définissons une partition $\pi_{i_1 i_2 \ldots i_r}$ sur l'ensemble A_N en disant que $a_m \equiv a_p\ (\pi_{i_1 i_2 \ldots i_r})$ si et seulement si l'on a $a_{m_j} = a_{p_j}$ pour tout $j = i_1, i_2, \ldots, i_r$. Ainsi donc, deux

états a_m, a_p de l'automate résultant tombent dans un même ensemble de $\pi_{i_1 i_2 \ldots i_r}$ si et seulement si dans leurs codes les composantes $i_1, i_2, \ldots, i_r$ sont respectivement égales.

EXEMPLE. Reprenons l'automate résultant S_N dont les états sont codés comme il est montré dans le tableau 4-27. De ce tableau on tire immédiatement que

$$\pi_{12} = \overline{1.2},\ \overline{3.4},\ \overline{5.6},\ \overline{7.8},\ \overline{9.10},\ \overline{11.12};$$
$$\pi_{23} = \overline{1.5.9},\ \overline{2.6.10},\ \overline{3.7.11},\ \overline{4.8.12}.$$

On voit sans peine que chaque ensemble de $\pi_{i_1 i_2 \ldots i_r}$ correspond à des éléments différents de l'ensemble $A_{i_1} \times A_{i_2} \times \ldots \times A_{i_r}$ (où A_{i_j} est l'ensemble d'états de l'automate composant S_{i_j}, $j = 1, \ldots, r$); autrement dit, la partition $\pi_{i_1 i_2 \ldots i_r}$ contient autant d'ensembles qu'il y a d'états internes différents dans le système d'automates $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$. En effet, la partition π_{23} admet 4 ensembles, ce qui correspond au nombre d'états du système d'automates constitué par S_2 et S_3: $C \times D = \{(c_1, d_1), (c_1, d_2), (c_2, d_1), (c_2, d_2)\}$. L'ensemble $\overline{1.5.9}$ de π_{23} correspond à l'état (c_1, d_1) du système S_2 et S_3; les états a_1, a_5, a_9 de S_N admettent les mêmes composantes, c_1 et d_1 dans les positions 2 et 3. L'ensemble $\overline{2.6.10}$ correspond à l'état (c_1, d_2), l'ensemble $\overline{3.7.11}$ à l'état (c_2, d_1), et l'ensemble $\overline{4.8.12}$ à l'état (c_2, d_2).

Si dans $\pi_{i_1 i_2 \ldots i_r}$ on a $r = 1$, il est possible de définir sur A_N une partition π_i, appelée quelquefois *partition primaire*. De toute évidence, $a_m \equiv a_p\ (\pi_i)$ si et seulement si on a $a_{m_i} = a_{p_i}$ ($a_m, a_p \in A_N$; $a_{m_i}, a_{p_i} \in A_i$). Ainsi donc, dans le même ensemble de la partition π_i tombent les états de l'automate résultant dont les i-ièmes composantes sont identiques.

Dans notre exemple

$$\pi_1 = \overline{1.2.3.4}\ \ \overline{5.6.7.8},\ \overline{9.10.11.12};$$
$$\pi_2 = \overline{1.2.5.6.9.10},\ \overline{3.4.7.8.11.12};$$
$$\pi_3 = \overline{1.3.5.7.9.11},\ \overline{2.4.6.8.10.12}.$$

Il est clair que le nombre des ensembles de la partition π_i est égal au nombre des états de l'automate composant S_i et qu'il existe une correspondance biunivoque entre les ensembles de π_i et les états de S_i.

Dans notre exemple

π_1, B_1: $\overline{1.2.3.4} \to b_1$, $\overline{5.6.7.8} \to b_2$, $\overline{9.10.11.12} \to b_3$;

π_2, B_2: $\overline{1.2.5.6.9.10} \to c_1$, $\overline{3.4.7.8.11.12} \to c_2$;

π_3, B_3: $\overline{1.3.5.7.9.11} \to d_1$, $\overline{2.4.6.8.10.12} \to d_2$.

On peut désormais identifier les états de S_i aux ensembles de la partition π_i en admettant que l'automate S_1, par exemple, présente trois états : $\overline{1.2.3.4}$, $\overline{5.6.7.8}$ et $\overline{9.10.11.12}$. Autrement dit,

$$\pi_1 = \overline{1.2.3.4},\ \overline{5.6.7.8},\ \overline{9.10.11.12} = \{b_1\ \ b_2,\ b_3\} = B,$$
$$\pi_2 = \overline{1.2.5.6.9.10},\ \overline{3.4.7.8.11.12} = \{c_1,\ c_2\} = C,$$
$$\pi_3 = \overline{1.3.5.7.9.11},\ \overline{2.4.6.8.10.12} = \{d_1,\ d_2\} = D.$$

4-5. Partitions à propriété de substitution

Une partition π sur l'ensemble d'états internes d'un automate $S = (A, Z, W, \delta, \lambda)$ possède la *propriété de substitution* (est une partition p.s.) si et seulement si de $a_m \equiv a_s\ (\pi)$ (où a_m et a_s sont contenus dans un même ensemble de π) il ressort que $\delta\ (a_m, z_f) \equiv \equiv \delta\ (a_s, z_f)\ (\pi)$ pour tout $z_f \in Z$.

Autrement dit, à partir de tous les états appartenant à un même ensemble de π l'automate passe, sous l'action de n'importe quel signal d'entrée, à un des états tels qu'ils appartiennent tous aussi à un même ensemble de π. Cela veut dire que tout signal d'entrée réalise une application entre ensembles de π. Ainsi donc, pour $z_f \in Z$ et $B \in \pi$, il existe un seul ensemble $B' \in \pi$ tel que $\delta\ (B, z_f) \subseteq B'$.

Exemple. L'automate S dont la table des transitions est représentée par le tableau 4-37 admet deux partitions p.s. : $\pi_1 = \overline{1.2.3}$, $\overline{4.5.6}$ et $\pi_2 = \overline{1.6}$, $\overline{2.5}$, $\overline{3.4}$.

Tableau 4-37

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_4	a_6	a_5	a_2	a_1	a_3
z_2	a_3	a_3	a_2	a_5	a_4	a_4

Vérifions-le pour π_1. Cherchons les transitions provoquées par z_1 et z_2 à partir de tous les états des deux ensembles de π:

$\overline{1.2.3}$:

$$\delta(a_1, z_1) = a_4, \qquad \delta(a_1, z_2) = a_3,$$
$$\delta(a_2, z_1) = a_6, \qquad \delta(a_2, z_2) = a_3,$$
$$\delta(a_3, z_1) = a_5, \qquad \delta(a_3, z_2) = a_2,$$
$$\delta(\overline{1.2.3}, z_1) = \overline{4.5.6}; \qquad \delta(\overline{1.2.3}, z_2) \subseteq \overline{1.2.3}.$$

$\overline{4.5.6}$:

$$\delta(a_4, z_1) = a_2, \qquad \delta(a_4, z_2) = a_5,$$
$$\delta(a_5, z_1) = a_1, \qquad \delta(a_5, z_2) = a_4,$$
$$\delta(a_6, z_1) = a_3, \qquad \delta(a_6, z_2) = a_4,$$
$$\delta(\overline{4.5.6}, z_1) = \overline{1.2.3}; \qquad \delta(\overline{4.5.6}, z_2) \subseteq \overline{4.5.6}.$$

Puisque les ensembles d'une partition p.s. π définie sur l'ensemble d'états A de S sont envoyés par n'importe quel signal d'entrée dans des ensembles de cette même partition, on peut les assimiler aux états d'un automate sans sortie (semi-automate) défini par π et S.

Soit π une partition p.s. sur l'ensemble d'états de l'automate $S = (A, Z, W, \delta, \lambda)$. On appelle *$\pi$-image* de S un semi-automate $S_\pi = (\{B_\pi\}, Z, \delta_\pi)$, $B_\pi \in \pi$, tel que $\delta_\pi(B_\pi, z_f) = B'_\pi$ si et seulement si $\delta(B_\pi, z_f) \subseteq B_\pi$.

Exemple. L'automate $S_{\pi_1} = (\pi_1, Z, \delta_{\pi_1})$ défini dans le tableau 4-38 est une π-image de l'automate S du tableau 4-37 pour la partition $\pi_1 = \overline{1.2.3}$, $\overline{4.5.6} = \{b_1, b_2\}$. L'automate $S_{\pi_2} = (\pi_2, Z, \delta_{\pi_2})$ défini dans le tableau 4-39 est une π-image du même automate S pour $\pi_2 = \overline{1.6}, \overline{2.5}, \overline{3.4} = \{c_1, c_2, c_3\}$.

Tableau 4-38

$\delta_{\pi_1}: \pi_1 \times Z \to \pi_1$

	b_1	b_2
z_1	b_2	b_1
z_2	b_1	b_2

Tableau 4-39

$\delta_{\pi_2}: \pi_2 \times Z \to \pi_2$

	c_1	c_2	c_3
z_1	c_3	c_1	c_2
z_2	c_3	c_3	c_2

De toute évidence, les partitions p.s. triviales sont $\pi(0)$ et $\pi(1)$.

Pour donner une idée de l'utilisation des partitions pour la décomposition des automates, examinons le travail en commun des

semi-automates S_{π_1} et S_{π_2} (tableaux 4-38 et 4-39). Chacun de ces semi-automates fait une partie du travail accompli par l'automate S, puisqu'ils ne font que définir les ensembles de π_1 ou de π_2 qui contiennent tel ou tel état de S. Or, fonctionnant ensemble, ils individualisent tout état de S, étant donné que l'intersection de deux ensembles quelconques de π_1 et π_2 contient un état et n'en contient qu'un seul. Ainsi donc, les états de S_{π_1} et S_{π_2} définissent sans erreur possible l'état correspondant de S; par conséquent, ensemble, les automates S_{π_1} et S_{π_2} calculent les transitions de l'automate S.

On montre sans peine que si π_1 et π_2 sont deux partitions p.s., alors $\pi_1\pi_2$ et $\pi_1 + \pi_2$ sont aussi deux partitions p.s. [28].

La procédure de recherche de toutes les partitions p.s. pour un automate donné est suffisamment simple, bien qu'elle exige souvent des calculs fastidieux. Elle s'opère en deux étapes [28]:

1. On calcule, pour chaque couple d'états (a_m, a_s), la plus petite partition p.s. (a_m, a_s) identifiant a_m à a_s (partitions p.s. primaires).

2. On cherche toutes les sommes possibles des $\pi\ (a_m, a_s)$ obtenues à la première étape : elles forment des partitions p.s. secondaires.

Identifions deux états a_m et a_s $(a_m, a_s \in A)$ contenus dans un même ensemble de la partition π cherchée : $a_m \equiv a_s\ (\pi)$. De la définition d'une partition à propriété de substitution il ressort alors que pour tout $z_f \in Z$ les états $\delta\ (a_m, z_f)$ et $\delta\ (a_s, z_f)$ doivent eux aussi être identifiés : $\delta\ (a_m, z_f) \equiv \delta\ (a_s, z_f)\ (\pi)$. Il est clair que si un état a_k est identifié à a_l et a_l à a_r, on devra avoir $a_k \equiv a_r\ (\pi)$, car la partition définit d'une façon univoque la relation d'équivalence qui, elle, est transitive. On applique cette procédure à chaque couple d'états figurant dans le même ensemble, jusqu'à ce que l'identification des nouveaux états cesse d'être possible. On remarque facilement que la partition $\pi\ (a_m, a_s)$ ainsi construite possède la propriété de substitution et représente la plus petite partition qui puisse identifier les états a_m et a_s dans le même ensemble. Pour construire d'autres partitions, on applique la procédure décrite à chaque couple d'états possible, c'est-à-dire $M\ (M - 1)/2$ fois, où M est le nombre des états de l'automate.

Tableau 4-40

$\delta : A \times Z \rightarrow A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_5	a_6	a_5	a_5	a_5	a_5
z_2	a_3	a_3	a_1	a_6	a_4	a_4
z_3	a_2	a_1	a_1	a_2	a_2	a_1

A la deuxième étape on cherche les partitions p.s. sans consulter la table des transitions, en sommant les partitions p.s. minimales et en faisant les sommes des partitions p.s. nouvellement obtenues avec les partitions p.s. déjà existantes.

Tableau 4-41

Procédure de construction de partitions p.s.

Couples identifiés	Couples essayés z_1	z_2	z_3	Partitions p.s. résultantes	Partition p.s.
1.2	5.6	3	1.2	$\overline{1.2}, \bar{3}, \bar{4}, \overline{5.6}$	π_1
5.6	5	4	1.2		
1.3	5	1.3	1.2	$\overline{1.2.3}, \bar{4}, \overline{5.6}$	π_2
1.2	5.6	3	1.2		
5.6	5	4	1.2		
1.4	5	3.6	2	$\overline{1.4}, \bar{2}, \overline{3.6}, \bar{5}$	π_3
3.6	5	1.4	1		
1.5	5	3.4	2	$\overline{1.2.5.6}, \overline{3.4}$	π_4
3.4	5	1.6	1.2		
1.6	5	3.4	1.2		
1.2	5.6	3	1.2		
5.6	5	4	1.2		
1.6	5	3.4	1.2	$\overline{1.2.5.6}, \overline{3.4}$	π_4
3.4	5	1.6	1.2		
1.2	5.6	3	1.2		
5.6	5	4	1.2		
2.3	5.6	1.3	1	$\overline{1.2.3}, \bar{4}, \overline{5.6}$	π_2
5.6	5	4	1.2		
1.3	5	1.3	1.2		
1.2	5.6	3	1.2		
2.4	5.6	3.6	1.2	$\overline{1.2.4}, \overline{3.5.6}$	π_5
5.6	5	4	1.2		
3.6	5	1.4	1		
1.4	5	3.6	2		
1.2	5.6	3	1.2		
2.5	5.6	3.4	1	$\overline{1.2.5.6}, \overline{3.4}$	π_4
5.6	5	4	1.2		
3.4	5	1.6	1.2		
1.2	5.6	3	1.2		
1.6	5	3.4	1.2		
2.6	5.6	3.4	1	$\overline{1.2.5.6}, \overline{3.4}$	π_4
5.6	5	4	1.2		
3.4	5	1.6	1.2		
1.2	5.6	3	1.2		
1.6	5	3.4	1.2		
3.4	5	1.6	1.2	$\overline{1.2.5.6}, \overline{3.4}$	π_4
1.6	5	3.4	1.2		
1.2	5.6	3	1.2		
5.6	5	4	1.2		
3.5	5	1.4	1.2	$\overline{1.2.4}, \overline{3.5.6}$	π_5
1.4	5	3.6	2		
1.2	5.6	3	1.2		
3.6	5	1.4	1		
5.6	5	4	1.2		
3.6	5	1.4	1	$\overline{1.4}, \bar{2}, \overline{3.6}, \bar{5}$	π_3
1.4	5	3.6	2		
4.5	5	4.6	2	$\overline{1.2}, \bar{3}, \overline{4.5.6}$	π_6
4.6	5	4.6	1.2		
1.2	5.6	3	1.2		
5.6	5	4	1.2		
4.6	5	4.6	1.2	$\overline{1.2}, \bar{3}, \overline{4.5.6}$	π_6
1.2	5.6	3	1.2		
5.6	5	4	1.2		
5.6	5	4	1.2	$\overline{1.2}, \bar{3}, \bar{4}, \overline{5.6}$	π_1
1.2	5.6	3	1.2		

EXEMPLE. Soit un automate défini par la table des transitions du tableau 4-40 *). Ses partitions p.s. triviales sont

$$\pi(0) = \overline{1},\ \overline{2},\ \overline{3},\ \overline{4},\ \overline{5},\ \overline{6};\ \pi(1) = \overline{1.2.3.4.5.6}.$$

Identifions les états a_1 et a_2: on obtient $\overline{1.2}$, $\overline{3}$, $\overline{4}$, $\overline{5.6}$. De la table des transitions on voit que les éléments de $\overline{5.6}$ doivent être identifiés. Or, $\overline{5.6}$ ne nécessite plus aucune nouvelle identification de couples; on a donc $\pi(1, 2) = \overline{1.2}, \overline{3}, \overline{4}, \overline{5.6}$.

Tous les calculs effectués se laissent facilement résumer par un tableau (voir tableau 4-41). On obtient de la sorte six partitions p.s. primaires. En faisant leurs sommes de toutes les façons possibles, on n'arrive à dégager qu'une seule nouvelle partition p.s. $\pi_7 = \pi_2 + \pi_6 = \overline{1.2.3}, \overline{4}, \overline{5.6} + \overline{1.2}, \overline{3}, \overline{4.5.6} = \overline{1.2.3}, \overline{4.5.6}$. En faisant les sommes de π_7 avec $\pi_1, \ldots, \pi_6$, on n'obtient aucune nouvelle partition p.s. Ainsi donc, l'automate défini par le tableau 4-40 admet neuf partitions p.s.: $\pi(0)$, $\pi(1)$, $\pi_1, \ldots, \pi_7$.

4-6. Couples de partitions

Deux partitions (π, π') définies sur l'ensemble d'états A de l'automate $S = (A, Z, W, \delta, \lambda)$ sont appelées *couple de partitions* si et seulement si de $a_m \equiv a_s\ (\pi)$ il ressort que $\delta(a_m, z_f) \equiv \delta(a_s, z_f)\ (\pi')$ pour tout $z_f \in Z$; $a_m, a_s \in A$.

Tableau 4-42

δ: $A \times Z \to A$

	a_1	a_2	a_3	a_4
z_1	a_1	a_3	a_2	a_4
z_2	a_2	a_4	a_1	a_3
z_3	a_3	a_1	a_4	a_2

Autrement dit, (π, π') est un couple de partitions si n'importe quel signal d'entrée sollicitant S applique des ensembles de π dans des ensembles de π', c'est-à-dire si pour chaque $z_f \in Z$ et $B \in \pi$ il existe un ensemble et un seul $B' \in \pi'$ tel que $\delta(B, z_f) \subseteq B'$.

EXEMPLE. Soient $\pi_1 = \overline{1.2}, \overline{3.4}$ et $\pi_2 = \overline{1.3}, \overline{2.4}$ deux partitions sur l'ensemble d'états de l'automate S dont la table des transitions est donnée dans le tableau 4-42. On voit sans peine que (π_1, π_2) est un couple de partitions. En effet,

$\overline{1.2}$:

$$\delta(a_1, z_1) = a_1, \quad \delta(a_1, z_2) = a_2, \quad \delta(a_1, z_3) = a_3,$$
$$\delta(a_2, z_1) = a_3, \quad \delta(a_2, z_2) = a_4, \quad \delta(a_2, z_3) = a_1,$$
$$\delta(\overline{1.2}, z_1) = \overline{1.3}; \quad \delta(\overline{1.2}, z_2) = \overline{2.4}; \quad \delta(\overline{1.2}, z_2) = \overline{1.3}.$$

*) Cet exemple est tiré de [14].

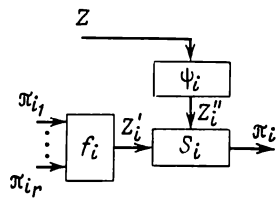
$\overline{3.4}$:

$$\delta(a_3, z_1) = a_2, \quad \delta(a_3 \; z_2) = a_1, \quad \delta(a_3, z_3) = a_4,$$
$$\delta(a_4, z_1) = a_4, \quad \delta(a_4, z_2) = a_3, \quad \delta(a_4, z_3) = a_2,$$
$$\delta(\overline{3.4}, z_1) = \overline{2.4}; \quad \delta(\overline{3.4}, z_2) = \overline{1.3}; \quad \delta(\overline{3.4}, z_3) = \overline{2.4}.$$

Il est clair que si π est une partition p.s., alors (π, π) est un couple de partitions. C'est ainsi que dans le dernier exemple $\pi_3 = \overline{1.4}, \overline{2.3}$ est une partition p.s., aussi $(\pi_3, \pi_3) = (\overline{1.4}, \overline{2.3}; \overline{1.4}, \overline{2.3})$ forment-elles un couple de partitions.

On montre que si (π, π') et (τ, τ') sont deux couples de partitions sur l'ensemble d'états A de l'automate S, alors $(\pi\tau, \pi'\tau')$ et $(\pi + \tau, \pi' + \tau')$ sont également des couples de partitions sur A [28].

Remarquons que pour une partition quelconque π définie sur l'ensemble d'états de S les couples de partitions $(\pi, \pi(1))$ et $(\pi(0), \pi)$ sont triviaux. En outre, si (π, π') est un couple de partitions et si $\eta \leqslant \pi$, alors (η, π') est aussi un couple de partitions. De même, si $\tau \geqslant \pi'$, alors (π, τ) est aussi un couple de partitions.



Fir. 4-11. Automate composant S_i

Les partitions p.s. et les couples de partitions définis sur l'ensemble d'états d'un automate jouent un grand rôle dans la théorie de décomposition des automates. Supposons par exemple que les sorties des automates composants $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$ sont appliquées sur l'entrée d'un automate composant S_i (fig. 4-11). On a montré plus haut que le nombre d'ensembles d'une partition π_i est égal au nombre d'états de l'automate composant S_i et qu'il existe une correspondance biunivoque entre les ensembles de π_i et les états de S_i. D'une façon analogue, une partition $\pi_{i_1 i_2 \ldots i_r}$ définit les signaux d'entrée appliqués à f_i en provenance des autres automates, car il existe une correspondance biunivoque entre les ensembles de $\pi_{i_1 i_2 \ldots i_r}$ et les états du système d'automates $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$. Comme dans n'importe quel automate, l'état suivant de S_i (l'ensemble de la partition π_i) est défini par son état actuel et le signal d'entrée (constitué en l'occurrence par l'état du système d'automates $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$ et la lettre de l'alphabet d'entrée externe $z_f \in Z_i''$). Ainsi donc, l'état de S_i se définit à chaque instant par l'état du système d'automates $S_{i_1}, S_{i_2}, \ldots, S_{i_r}, S_i$ et par $z_f \in Z_i''$, c'est-à-dire par un ensemble de la partition $\pi_{i_1 i_2 \ldots i_r i}$ et par $z_f \in Z_i''$; pour cette raison $(\pi_{i_1 i_2 \ldots i_r i}, \pi_i)$ est un couple de partitions, et δ_i réalise une application $\pi_{i_1 i_2 \ldots i_r i} \times Z_i$ dans π_i.

Remarquons aussi que si l'entrée de S_i n'est pas reliée aux sorties des autres automates composants (ce qui signifie que S_i reçoit uni-

quement des signaux de Z_i''), alors δ_i réalise une application $\pi_i \times Z_i''$ dans π_i ; cela veut dire que (π_i, π_i) est un couple de partitions et que, par conséquent, π_i est une partition p.s.

EXEMPLE. En considérant le réseau montré sur la figure 4-9, on voit que π_1 doit être une partition p.s. et que (π_{123}, π_2) et (π_{23}, π_3) sont des couples de partitions. Nous laissons au lecteur le soin de le vérifier à l'aide du tableau 4-28.

4-7. Théorème général de décomposition

Le théorème général de décomposition des automates est exposé dans [35]. De sa démonstration se dégage un procédé avantageux de construction d'un réseau d'automates dont l'automate résultant réalise un automate donné. Nous allons formuler ce théorème, évoquer les idées fondamentales utilisées pour sa démonstration et décrire le procédé pratique de la décomposition.

Soit $\{\pi_i\}$, $1 \leqslant i \leqslant n$, un ensemble de partitions sur l'ensemble d'états A de l'automate à décomposer $S = (A, Z, W, \delta, \lambda)$.

THÉORÈME. *Pour un ensemble de partitions* $\{\pi_i\}$, $1 \leqslant i \leqslant n$, *il existe un réseau abstrait d'automates* N *tel que* $R(S) = S_N$, *si et seulement si*

$$\prod_{i=1}^{n} \pi_i = \pi(0). \tag{4-1}$$

Les partitions π_i *et les automates composants* S_i *sont liés alors par une correspondance biunivoque.*

Un ensemble de partitions vérifiant la condition (4-1) sera appelé *orthogonal*.

Afin d'illustrer la méthode de décomposition décrite dans la démonstration du théorème (voir [35]), nous allons décomposer l'automate de Mealy $S = (A, Z, W, \delta, \lambda)$ défini dans les tableaux 4-43 et 4-44.

Tableau 4-43

$\delta : A \times Z \to A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_2	a_5	a_1	a_6	a_1	a_2
z_2	a_6	a_1	a_5	a_2	a_1	a_6
z_3	a_6	a_1	a_5	a_2	a_2	a_5
z_4	a_2	a_5	a_1	a_6	a_4	a_3

Tableau 4-44

$\lambda : A \times Z \to W$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	w_1	w_2	w_1	w_1	w_3	w_2
z_2	w_2	w_1	w_3	w_2	w_1	w_2
z_3	w_1	w_1	w_1	w_2	w_3	w_3
z_4	w_3	w_3	w_1	w_3	w_1	w_1

Soient ses trois partitions

$$\pi_1 = \overline{1.2.3.4},\ \overline{5.6};\ \pi_2 = \overline{1.2.5.6},\ \overline{3.4};\ \pi_3 = \overline{1.3.5},\ \overline{2.4.6},$$

telles que

$$\pi_1\pi_2\pi_3 = \pi\ (0).$$

Nous associons à chaque partition π_i une fonction $F_i : A \times Z \to \pi_i$ telle que $F_i\ (a_m,\ z_f) = \pi_i\ (\delta\ (a_m,\ z_f))$, c'est-à-dire que sa valeur sur le couple $(a_m,\ z_f)$ soit égale à l'ensemble de π_i qui contient l'état $a_s = \delta\ (a_m,\ z_f)$; $a_m,\ a_s \in A,\ z_f \in Z$.

Tableau 4-45

F_1: $A \times Z \to B$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	b_1	b_2	b_1	b_2	b_1	b_1
z_2	b_2	b_1	b_2	b_1	b_1	b_2
z_3	b_2	b_1	b_2	b_1	b_1	b_2
z_4	b_1	b_2	b_1	b_2	b_1	b_1

Tableau 4-46

F_2: $A \times Z \to C$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	c_1	c_1	c_1	c_1	c_1	c_1
z_2	c_1	c_1	c_1	c_1	c_1	c_1
z_3	c_1	c_1	c_1	c_1	c_1	c_1
z_4	c_1	c_1	c_1	c_1	c_2	c_2

Les fonctions F_1, F_2 et F_3 sont définies dans les tableaux 4-45, 4-46 et 4-47 respectivement. Elles se déduisent de la fonction $\delta : A \times Z \to A$ (tableau 4-43) en remplaçant les états a_s par l'ensemble de π_i qui contient a_s. Nous adopterons les notations suivantes pour désigner les ensembles des partitions π_1, π_2 et π_3:

Tableau 4-47

F_3: $A \times Z \to D$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	d_2	d_1	d_1	d_2	d_1	d_2
z_2	d_2	d_1	d_1	d_2	d_1	d_2
z_3	d_2	d_1	d_1	d_2	d_2	d_1
z_4	d_2	d_1	d_1	d_2	d_2	d_1

$$\pi_1 = \overline{1.2.3.4},\ \overline{5.6} = \{b_1,\ b_2\} = B,$$

$$\pi_2 = \overline{1.2.5.6},\ \overline{3.4} = \{c_1,\ c_2\} = C,$$

$$\pi_3 = \overline{1.3.5},\ \overline{2.4.6} = \{d_1,\ d_2\} = D.$$

Faisons sur A et Z des partitions τ_i et η_i respectivement, de telle sorte que:

1) deux états a_m, a_s tombent dans un même ensemble de la partition τ_i $(a_m \equiv a_s\ (\tau_i))$ si et seulement si pour tout $z_f \in Z$ on a

$$F_i\ (a_m,\ z_f) = F_i\ (a_s,\ z_f),$$

c'est-à-dire qu'on réunit dans chaque ensemble de τ_i les états auxquels correspondent des colonnes identiques du tableau de F_i;

2) deux signaux d'entrée z_f, z_t tombent dans un même ensemble de la partition η_i ($z_f \equiv z_t\ (\eta_i)$) si et seulement si pour tout $a_m \in A$ on a

$$F_i\ (a_m,\ z_f) = F_i\ (a_m,\ z_t),$$

c'est-à-dire qu'on réunit dans chaque ensemble de η_i les signaux d'entrée auxquels correspondent les lignes identiques du tableau de F_i.

On obtient de la sorte

$$\tau_1 = \overline{1.3.6},\ \overline{2.4},\ \overline{5};\qquad \eta_1 = \overline{z_1 z_4},\ \overline{z_2 z_3};$$
$$\tau_2 = \overline{1.2.3.4},\ \overline{5.6};\qquad \eta_2 = \overline{z_1 z_2 z_3},\ \overline{z_4};$$
$$\tau_3 = \overline{1.4},\ \overline{2.3},\ \overline{5},\ \overline{6};\qquad \eta_3 = \overline{z_1 z_2},\ \overline{z_3 z_4}.$$

Il est facile de voir qu'en assortissant (τ_i, π_i), on obtient des couples de partitions, c'est-à-dire que n'importe quel signal d'entrée envoie chaque ensemble de π_i sur un ensemble de τ_i. On remarque en outre que τ_i est la partition maximale pour le couple (τ_i, π_i); cela ressort de la construction même de τ_i, dont les ensembles regroupent les colonnes identiques du tableau de la fonction F_i.

Construisons le réseau $N = (Z_N,\ \{S_i\},\ W_N,\ \{f_i\},\ \{\psi_i\},\ g)$. A cet effet, cherchons toutes les composantes du cortège N *). Commençons par les alphabets d'entrée et de sortie du réseau.

1. Posons $Z_N = Z$.
2. Posons $W_N = W$.

Dans notre exemple, $Z_N = \{z_1,\ z_2,\ z_3,\ z_4\}$, $W_N = \{w_1,\ w_2,\ w_3\}$.

3. Construisons les automates composants $S_i = (A_i,\ Z_i,\ \delta_i)$, $1 \leqslant i \leqslant n$ (la base du réseau):

a) Posons $A_i = \pi_i$. Dans notre exemple, cela donne $A_1 = \{b_1,\ b_2\} = B$, $A_2 = \{c_1,\ c_2\} = C$, $A_3 = \{d_1,\ d_2\} = D$.

b) Pour composer l'alphabet d'entrée d'un automate Z_i, nous utiliserons les partitions τ_i et η_i introduites plus haut. Rappelons qu'on a

$$Z_i = \begin{cases} Z_i' \times Z_i'' & \text{pour } Z_i' \neq \varnothing, \\ Z_i'' & \text{pour } Z_i' = \varnothing. \end{cases} \tag{4-2}$$

Ici Z_i', Z_i'' sont respectivement les alphabets d'entrée interne et externe de l'automate S_i.

Nous avons vu ci-dessus que (τ_i, π_i) est un couple de partitions et que τ_i est la plus grande partition qui puisse être assortie à π_i.

*) Les alphabets d'entrée et de sortie du réseau sont désignés par les notations Z_N et W_N, les mêmes que pour l'automate résultant du réseau, afin de les distinguer des alphabets d'entrée Z et de sortie W de l'automate initial $S = (A,\ Z,\ W,\ \delta,\ \lambda)$ qu'il s'agit de décomposer.

D'autre part, on a vu dans le paragraphe 4-6 que si l'entrée de f est liée aux sorties $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_r}$ des automates composants $S_{i_1}, S_{i_2}, \ldots, S_{i_r}$ (fig. 4-11), alors $(\pi_{i_1 i_2 \ldots i_r i}, \pi_i)$ est un couple de partitions dans lequel $\pi_{i_1 i_2 \ldots i_r i} \leqslant \tau_i$, parce que τ_i est la partition maximale pour π_i. On montre tout aussi facilement que $\pi_{i_1 i_2 \ldots i_r i} = \pi_{i_1}\pi_{i_2} \ldots \pi_{i_r}\pi_i$. Ainsi donc, pour reconstituer les automates dont les sorties aboutissent à l'entrée de f_i, on doit chercher un produit

$$\pi_{i_1}\pi_{i_2} \ldots \pi_{i_r}\pi_i = \pi_i \prod_{j=i_1}^{i_r} \pi_j$$

non supérieur à τ_i *).

Dans notre exemple, pour avoir le produit indiqué, nous chercherons d'abord tous les produits possibles des partitions deux à deux $\pi_i\pi_j$:

$$\pi_1\pi_2 = \overline{1.2},\ \overline{3.4},\ \overline{5.6};\quad \pi_1\pi_3 = \overline{1.3},\ \overline{2.4},\ \overline{5},\ \overline{6};$$
$$\pi_2\pi_3 = \overline{1.5},\ \overline{2.6},\ \overline{3},\ \overline{4}.$$

Rappelons en outre que les partitions π_1, π_2 et π_3 ont été choisies de façon à vérifier la condition $\pi_1\pi_2\pi_3 = \pi\ (0)$.

Comparant ces produits avec τ_1, τ_2 et τ_3, on s'assure que $\pi_1\pi_3 \leqslant \tau_1$; $\pi_1 \leqslant \tau_2$; $\pi_1\pi_2\pi_3 \leqslant \tau_3$.

Introduisons une partition ξ_i telle que

$$\xi_i = \prod_{j=i_1}^{i_r} \pi_j, \quad j \neq i;$$

ce produit ne contient pas π_i, car on peut relier à l'entrée de f_i seules les sorties des automates composants, distincts de S_i (voir fig. 4-11).

Dans notre exemple

$$\xi_1 = \pi_3 = \overline{1.3.5},\ \overline{2.4.6};\quad \xi_2 = \pi_1 = \overline{1.2.3.4},\ \overline{5.6};$$
$$\xi_3 = \pi_1\pi_2 = \overline{1.2},\ \overline{3.4},\ \overline{5.6}.$$

Dans l'automate S_i nous posons $Z_i' = \xi_i$, $Z_i'' = \eta_i$ et nous définissons Z_i par (4-2).

c) Cherchons la fonction des transitions de S_i : $\delta_i : \pi_i \times Z_i \to \pi_i$.

Soient α, β et γ des ensembles des partitions π_i, ξ_i et η_i respectivement ($\alpha \in \pi_i$, $\beta \in \xi_i$, $\gamma \in \eta_i$). Si $\xi_i = Z_i' = \varnothing$, c'est-à-dire si $\pi_i \leqslant \tau_i$ (π_i est une partition p.s.), alors

$$\delta_i(\alpha, \gamma) = B_{\pi_i}(\delta(\alpha, \gamma)).$$

*) L'existence d'un tel produit est corroborée par (4-1).

Ainsi donc, la valeur de la fonction des transitions δ_i est égale à celui des ensembles de la partition π_i qui contient $\delta(\alpha, \gamma)$, δ étant la fonction des transitions de l'automate à décomposer $S = (A, Z, W, \delta, \lambda)$.

Par contre, si $\xi_i = Z_i' \neq \varnothing$, alors

$$\delta_i(\alpha, (\beta, \gamma)) = \begin{cases} B_{\pi_i}(\delta(\alpha \cap \beta, \gamma)) \text{ si } \alpha \cap \beta \neq \varnothing; \\ \text{non définie, c'est-à-dire égale} \\ \quad \text{à un ensemble quelconque de } \pi_i, \\ \quad \text{si } \alpha \cap \beta = \varnothing. \end{cases}$$

Revenons à notre exemple et cherchons les fonctions des transitions pour chaque automate composant.

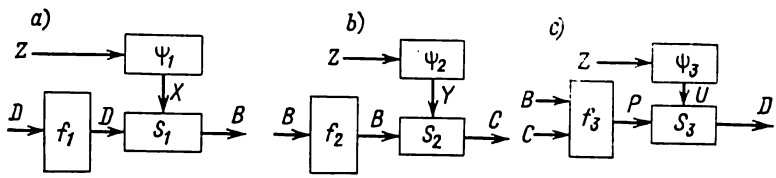


Fig. 4-12. Automates composants du réseau : S_1 (*a*), S_2 (*b*), S_3 (*c*)

Automate S_1 (fig. 4-12, *a*) :

$$\pi_1 = \overline{1.2.3.4},\ \overline{5.6} = \{b_1, b_2\} = B,$$
$$\eta_1 = \overline{z_1 z_4},\ \overline{z_2 z_3} = \{x_1, x_2\} = X,$$
$$\xi_1 = \pi_3 = \overline{1.3.5},\ \overline{2.4.6} = \{d_1, d_2\} = D.$$

La fonction $\delta_1 : B \times (D \times X) \to B$ est donnée dans le tableau 4-48. La même fonction décrite en termes de partitions est donnée dans le tableau 4-49.

Tableau 4-48

$\delta_1 : B \times (D \times X) \to B$

	b_1	b_2
$d_1 x_1$	b_1	b_1
$d_1 x_2$	b_2	b_1
$d_2 x_1$	b_2	b_1
$d_2 x_2$	b_1	b_2

Tableau 4-49

$\delta_1 : \pi_1 \times (\xi_1 \times \eta_1) \to \pi_1$

	$\overline{1.2.3.4}$	$\overline{5.6}$
$\overline{1.3.5}$, $\overline{z_1 z_4}$	$\overline{1.2.3.4}$	$\overline{1.2.3.4}$
$\overline{1.3.5}$, $\overline{z_2 z_3}$	$\overline{5.6}$	$\overline{1.2.3.4}$
$\overline{2.4.6}$, $\overline{z_1 z_4}$	$\overline{5.6}$	$\overline{1.2.3.4}$
$\overline{2.4.6}$, $\overline{z_2 z_3}$	$\overline{1.2.3.4}$	$\overline{5.6}$

Explicitons la construction de la fonction δ_1 en considérant la transition à partir de l'état b_1 sous l'action du signal d'entrée (d_1, x_2). Puisque $\xi_1 \neq \varnothing$, on a

$$\delta_1(b_1(d_1, x_2)) = B_{\pi_1}(\delta(b_1 \cap d_1, z_2)) =$$
$$= B_{\pi_1}(\delta(\overline{1.2.3.4} \cap \overline{1.3.5}, z_2)) = B_{\pi_1}(\delta(\overline{1.3}, z_2)) = B_{\pi_1}(a_6) = b_2.$$

Automate S_2 (fig. 4-12, b):

$$\pi_2 = \overline{1.2.5.6}, \ \overline{3.4} = \{c_1, c_2\} = C,$$
$$\eta_2 = \overline{z_1 z_2 z_3}, \ \overline{z_4} = \{y_1, y_2\} = Y,$$
$$\xi_2 = \pi_1 = \overline{1.2.3.4}, \ \overline{5.6} = \{b_1, b_2\} = B.$$

Tableau 4-50

δ_2: $C \times (B \times Y) \to C$

	c_1	c_2
b_1y_1	c_1	c_1
b_1y_2	c_1	c_1
b_2y_1	c_1	—
b_2y_2	c_2	—

Tableau 4-51

δ_3: $D \times (P \times U) \to D$

	d_1	d_2
p_1u_1	d_2	d_1
p_1u_2	d_2	d_1
p_2u_1	d_1	d_2
p_2u_2	d_1	d_2
p_3u_1	d_1	d_2
p_3u_2	d_2	d_1

Tableau 4-52

ω_3: $P \times U \to Q$

$P \times U$	Q
p_1u_1	q_1
p_1u_2	q_1
p_2u_1	q_2
p_2u_2	q_2
p_3u_1	q_2
p_3u_2	q_1

La fonction δ_2: $C \times (B \times Y) \to C$ est définie dans le tableau 4-50. Elle n'est pas définie sur les transitions $(c_2, (b_2, y_1))$ et $(c_2, (b_2, y_2))$, car $c_2 \cap b_2 = \overline{3.4} \cap \overline{5.6} = \varnothing$.

Tableau 4-53

δ_3: $D \times Q \to D$

	d_1	d_2
q_1	d_2	d_1
q_2	d_1	d_2

Automate S_3 (fig. 4-12, c):

$$\pi_3 = \overline{1.3.5}, \ \overline{2.4.6} = \{d_1, d_2\} = D,$$
$$\eta_3 = \overline{z_1 z_2}, \ \overline{z_3 z_4} = \{u_1, u_2\} = U,$$
$$\xi_3 = \overline{1.2}, \ \overline{3.4}, \ \overline{5.6} = \{p_1, p_2, p_3\} = P.$$

La fonction δ_3: $D \times (P \times U) \to D$ est définie dans le tableau 4-51. Puisque ce tableau présente des lignes identiques (la première, la deuxième et la sixième avec d_2, d_1, la troisième, la quatrième et la cinquième avec d_1, d_2), on peut, en introduisant encore un traducteur de fonction ω_3: $P \times U \to Q$ (tableau 4-52), comprimer la table des transitions jusqu'à deux lignes (tableau 4-53), auquel cas la structure de S_3 et de ses fonctions d'entrée prend la forme repré-

sentée sur la figure 4-13, *a*. Il est clair qu'on peut substituer aux trois traducteurs de fonction f_3, ψ_3 et ω_3 un traducteur unique φ_3: $B \times C \times Z \to Q$ (fig. 4-13, *b*). On pourrait bien sûr comprimer,

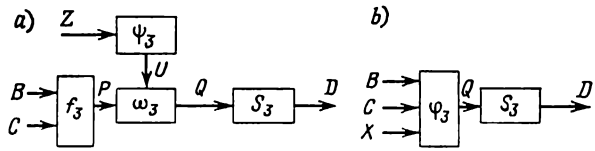


Fig. 4-13. Automate S_3 et ses fonctions d'entrée avec traducteurs de fonctions ω_3 (*a*) et φ_3 (*b*)

d'une façon analogue, aussi les tables des fonctions de transition δ_1, δ_2.

4. Cherchons les fonctions de connexion des automates composants :

$$f_i : \mathop{\times}_{j=i_1}^{i_r} A_j \to Z'_i,$$

qui peuvent s'écrire aussi en termes de partitions :

$$f_i : \mathop{\times}_{j=i_1}^{i_r} \pi_j \to \xi_i.$$

Soient $\pi_{i_1} \times \pi_{i_2} \times \ldots \times \pi_{i_r} = T_i$ *). Faisons un ensemble $T'_i \subseteq T_i$, $t_s \in T'_i$, $t_s = (t_{s_1}, \ldots, t_{s_j}, \ldots, t_{s_r})$, tel que

$$\bigcap_{j=1}^{r} t_{s_j} \neq \varnothing.$$

---

*) On prendra garde à ne pas confondre le produit

$$\pi_{i_1} \times \pi_{i_2} \times \ldots \times \pi_{i_r} = \mathop{\times}_{j=i_1}^{i_r} \pi_j,$$

qui est un produit d'ensembles, avec

$$\pi_{i_1} \times \pi_{i_2} \times \ldots \times \pi_{i_r} = \prod_{j=i_1}^{i_r} \pi_j,$$

qui est un produit de partitions. Dans le premier cas on obtient les vecteurs à r composantes, formés d'ensembles provenant chacun d'une partition π_j ($j = i_1, \ldots, i_r$). Dans le second cas, on obtient une partition unique dont les ensembles sont formés en faisant toutes les intersections possibles des ensembles des partitions $\pi_{i_1}, \pi_{i_2}, \ldots, \pi_{i_r}$ du type $B_{i_1} \cap B_{i_2} \cap \ldots \cap B_{i_r}$, où $B_j \in \pi_j$, $j = i_1, \ldots, i_r$ (nous en avons parlé en détail dans le § 4-4).

Seuls seront donc contenus dans T'_i les vecteurs de T_i pour lesquels l'intersection de toutes les composantes n'est pas vide. Une telle intersection

$$\bigcap_{j=1}^{r} t_{s_j}$$

existe, car les composantes $t_{s_1}, \ldots, t_{s_r}$ sont des ensembles de partitions.

La fonction f_i réalise l'application $T'_i \to \xi_i$. La valeur prise par f_i sera définie comme suit:

$$f_i(t_{s_1}, \ldots, t_{s_r}) = p_k \in \xi_i \text{ si } \bigcap_{j=1}^{r} t_{s_j} \subseteq p_k,$$

ce qui signifie que la valeur de f_i est égale à celui des ensembles de ξ_i qui renferme l'intersection des composantes $t_{s_1}, \ldots, t_{s_r}$.

Sur l'ensemble $T_i \setminus T'_i$ la fonction f_i n'est pas définie.

Dans notre exemple, nous avons pour l'automate composant S_3

$$T_3 = \pi_1 \times \pi_2 = \{\overline{1.2.3.4}, \overline{5.6}\} \times \{\overline{1.2.5.6}, \overline{3.4}\} =$$
$$= \{(\overline{1.2.3.4}, \overline{1.2.5.6}), (\overline{1.2.3.4}, \overline{3.4}), (\overline{5.6}, \overline{1.2.5.6}), (\overline{5.6}, \overline{3.4})\}.$$

Cherchons l'intersection des composantes dans des couples de T_3:

$$\overline{1.2.3.4} \cap \overline{1.2.5.6} = \overline{1.2}; \ \overline{1.2.3.4} \cap \overline{3.4} = \overline{3.4}; \ \overline{5.6} \cap \overline{1.2.5.6} = \overline{5.6};$$
$$\overline{5.6} \cap \overline{3.4} = \varnothing.$$

Alors

$$T'_3 = \{(\overline{1.2.3.4}, \overline{1.2.5.6}), (\overline{1.2.3.4}, \overline{3.4}), (\overline{5.6}, \overline{1.2.5.6})\}.$$

On a vu plus haut que $\xi_3 = \overline{1.2}, \overline{3.4}, \overline{5.6}$. La fonction de connexion f_3: $T'_3 \to \xi_3$ est définie dans le tableau 4-54. Puisque f_1 et f_2 admettent une seule entrée chacune, elles réalisent des applications identiques:

$$f_1: D \to D; \quad f_2: B \to B.$$

Tableau 4-54

f_3: $T'_3 \to \xi_3$

T'_3	ξ_3
$\overline{1.2.3.4}, \overline{1.2.5.6}$	$\overline{1.2}$
$\overline{1.2.3.4}, \overline{3.4}$	$\overline{3.4}$
$\overline{5.6}, \overline{1.2.5.6}$	$\overline{5.6}$

Tableau 4-55

Fonctions ψ_1, ψ_2, ψ_3

Z	ψ_1	ψ_2	ψ_3
z_1	x_1	y_1	u_1
z_2	x_2	y_1	u_1
z_3	x_3	y_1	u_2
z_4	x_4	y_2	u_2

5. L'ensemble des fonctions d'entrée sera défini comme suit:

$$\psi_i(z_f) = B_{\eta_i}(z_f), \quad i = 1, \ldots, n.$$

Cela veut dire que ψ_i prend sur $z_f \in Z$ la valeur égale à l'ensemble de η_i qui contient z_f. On conçoit donc que l'automate S_i ne discerne pas les lettres de l'alphabet d'entrée Z qui font partie d'un même ensemble de la partition η_i.

Dans notre exemple

$$\eta_1 = \overline{z_1 z_4},\ \overline{z_2 z_3} = \{x_1,\ x_2\}, \quad \eta_2 = \overline{z_1 z_2 z_3},\ \overline{z_4} = \{y_1\ y_2\},$$
$$\eta_3 = \overline{z_1 z_2},\ \overline{z_3 z_4} = \{u_1,\ u_2\}.$$

Les fonctions ψ_1, ψ_2 et ψ_3 sont résumées dans le tableau 4-55.

6. Construisons la fonction de sortie du réseau

$$g\colon \left(\mathop{\times}_{i=1}^{n} A_i\right) \times Z \to W;$$

elle s'écrit aussi en termes de partitions:

$$g\colon \left(\mathop{\times}_{i=1}^{n} \pi_i\right) \times Z \to W.$$

Soient $\pi_1 \times \pi_2 \times \ldots \times \pi_n = H$. Faisons un ensemble $H' \subseteq H$, $h_m \in H'$, $h_m = (h_{m_1}, \ldots, h_{m_i}, \ldots, h_{m_n})$, tel que

$$\bigcap_{i=1}^{n} h_{m_i} \neq \varnothing.$$

Seuls donc tombent dans H' les vecteurs de H pour lesquels l'intersection de toutes les composantes est non vide.

La fonction g réalise l'application $H' \times Z \to W$. Nous définirons sa valeur comme suit:

$$g((h_{m_1}, \ldots, h_{m_i}, \ldots, h_{m_n}),\ z_f) = \lambda\left(\bigcap_{i=1}^{n} h_{m_i},\ z_f\right).$$

Cela revient à dire que la valeur de la fonction de sortie du réseau se confond avec la valeur que prend la fonction de sortie λ de l'automate à décomposer S sur le couple $(a_m,\ z_f)$ dans lequel a_m est un état contenu dans l'intersection des composantes du vecteur $h_m \in H'$.

Sur l'ensemble $H \setminus H'$ la fonction g n'est pas définie.

Dans notre exemple

$$\pi_1 = \overline{1.2.3.4},\ \overline{5.6} = \{b_1,\ b_2\} = B,$$
$$\pi_2 = \overline{1.2.5.6},\ \overline{3.4} = \{c_1,\ c_2\} = C,$$
$$\pi_3 = \overline{1.3.5},\ \overline{2.4.6} = \{d_1,\ d_2\} = D;$$

$$H = B \times C \times D = \{(b_1,\ c_1,\ d_1),\ (b_1,\ c_1,\ d_2),\ (b_1,\ c_2,\ d_1),$$
$$(b_1,\ c_2,\ d_2),\ (b_2,\ c_1,\ d_1),\ (b_2,\ c_1,\ d_2),\ (b_2,\ c_2,\ d_1),\ (b_2,\ c_2,\ d_2)\}.$$

Par exemple,

$$g((b_1, c_1, d_1), z_1) = \lambda(b_1 \cap c_1 \cap d_1, z_1) =$$
$$= \lambda(\overline{1.2.3.4} \cap \overline{1.2.5.6} \cap \overline{1.3.5}, z_1) = \lambda(a_1, z_1) = w_1.$$

D'autre part,

$$g((b_2, c_2, d_1), z_1) = \lambda(b_2 \cap c_2 \cap d_1, z_1) =$$
$$= \lambda(\overline{5.6} \cap \overline{3.4} \cap \overline{1.3.5}, z_1) = \lambda(\varnothing, z_1),$$

ce qui revient à dire que la fonction g n'est pas définie sur le vecteur (b_2, c_2, d_1).

La fonction de sortie du réseau est donnée dans le tableau 4-56

Tableau 4-56

g: $(B \times C \times D) \times Z \to W$

	$b_1c_1d_1$	$b_1c_1d_2$	$b_1c_2d_1$	$b_1c_2d_2$	$b_2c_1d_1$	$b_2c_1d_2$
z_1	w_1	w_2	w_1	w_1	w_3	w_2
z_2	w_2	w_1	w_3	w_2	w_1	w_2
z_3	w_1	w_1	w_1	w_2	w_3	w_3
z_4	w_3	w_3	w_1	w_3	w_1	w_1

Le réseau N qu'on obtient par décomposition de l'automate S est représenté sur la figure 4-14.

Dans [35] il est montré qu'un réseau ainsi construit réalise l'automate initial S. En ce qui concerne la notion de réalisation d'un auto-

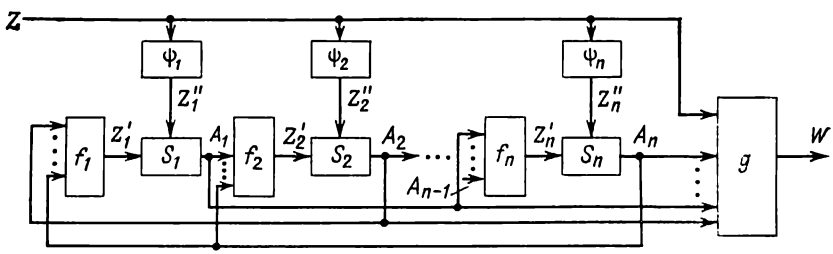


Fig. 4-14. Réseau N réalisant l'automate S

mate par un autre, nous l'avons définie plus haut. Nous dirons également que le réseau N est une réalisation de l'automate S si son automate résultant S_N réalise S ($S_N = R(S)$), c'est-à-dire si S_N admet un sous-automate S'_N isomorphe à S.

Le schéma de démonstration adopté dans [35] est le suivant. On construit, pour le réseau N, l'automate résultant $S_N = (A_N, Z_N, W_N, \delta_N, \lambda_N)$ et un sous-automate $S'_N = (A'_N, Z'_N, W'_N, \delta'_N, \lambda'_N)$ tel que

$$Z'_N = Z_N; \quad W'_N = W_N;$$

$$A'_N = \{(a_{m_1}, \ldots, a_{m_n}) \mid \bigcap_{i=1}^{n} a_{m_i} \neq \varnothing, \quad a_{m_i} \in A_i, \ (a_{m_1}, \ldots, a_{m_n}) \in A_N\},$$

$$\delta'_N((a_{m_1}, \ldots, a_{m_n}), z_f) = \delta_N((a_{m_1}, \ldots, a_{m_n}), z_f),$$

$$\lambda'_N((a_{m_1}, \ldots, a_{m_n}), z_f) = \lambda_N((a_{m_1}, \ldots, a_{m_n}), z_f).$$

Ceci fait, on montre que S'_N est un sous-automate de S_N et que S'_N est isomorphe à S ; par conséquent, $S_N = R(S)$, c'est-à-dire S_N réalise l'automate donné S.

Il ressort du procédé de construction du réseau N que les propriétés de ses automates composants S_i et leurs connexions dépendent de l'ensemble de partitions choisi $\{\pi_i\}$, $1 \leqslant i \leqslant n$. Le réseau aura autant d'automates composants qu'il y a de partitions dans $\{\pi_i\}$. Un automate composant S_i admet autant d'états internes qu'il y a d'ensembles dans la partition π_i. Puisque le nombre d'ensembles d'une partition non triviale π_i est toujours inférieur au nombre des éléments de l'ensemble d'états A de l'automate initial S, les automates composants ont toujours moins d'états que l'automate S.

Les partitions τ_i et η_i sont définies d'une façon univoque par la partition π_i (à l'aide de la fonction F_i) ; la partition τ_i désigne les automates qui agissent sur S_i, tandis que la partition η_i détermine les classes des lettres de l'alphabet d'entrée Z qui sont indiscernables pour l'automate S_i.

Le triplet (π_i, τ_i, η_i) s'appelle *triplet caractéristique* de l'automate S_i. Notons que τ_i et η_i sont des partitions maximales. Plus τ_i est grande, moins il y a d'automates dont les sorties agissent sur S_i. Plus η_i est grande, moins compliquée est la dépendance de δ_i par rapport à l'entrée externe Z. On voit donc que le recours aux partitions τ_i et η_i lors de la construction des S_i est nécessaire si l'on veut que le réseau N soit aussi simple que possible.

4-8. Matrice structurelle du réseau d'automates

Le procédé de construction pratique du réseau d'automates décrit dans le paragraphe précédent laisse voir que, dans le cas général, la structure d'un réseau N n'est pas définie de façon univoque, car l'inégalité

$$\prod_{j=i_1}^{i_r} \pi_j \leqslant \tau_i,$$

qui définit les automates $S_{i_1}, \ldots, S_{i_r}$ affectant le comportement de S_i, peut avoir lieu pour plusieurs collections de partitions faisant partie de l'ensemble $\{\pi_i\}$, $i = 1, \ldots, n$.

On appelle *matrice structurelle du réseau* N la matrice binaire carrée M_N de dimension n dont les colonnes correspondent aux partitions $\tau_1, \ldots, \tau_n$ et les lignes aux partitions $\pi_1, \ldots, \pi_n$. Dans la i-ième colonne de M_N on inscrit un 1 dans les lignes indiciées par $i_1, i_2, \ldots, i_r$ si et seulement si

$$\prod_{j=i_1}^{i_r} \pi_j \leqslant \tau_i. \tag{4-3}$$

Tableau 4-57

Matrice structurelle du réseau de la figure 4-14

	τ_1	τ_2	τ_3
π_1	1	1	1
π_2			1
π_3	1		1

Désignons par m_{ij} l'élément (i, j) de la matrice M_N. Alors, si $m_{ij} = 1$, l'automate S_i intervient dans le fonctionnement de l'automate S_j (en particulier, si $i \neq j$, la sortie de S_i est reliée à l'entrée de f_j).

La matrice structurelle du réseau de la figure 4-14 est donnée dans le tableau 4-57.

Il existe évidemment un grand nombre de matrices binaires différentes de dimension n qui vérifient la condition (4-3) et qui sont à ce titre les matrices structurelles du réseau N. Désignons par $\mathfrak{M}_N$ l'ensemble des matrices structurelles possibles de N. Evidemment $\mathfrak{M}_N \neq \varnothing$, car toute matrice dont les éléments sont tous égaux à 1 ($m_{ij} = 1$, $i, j = 1, \ldots, n$) est toujours une matrice structurelle. De cette façon $\mathfrak{M}_N$ détermine toutes les structures théoriquement possibles du réseau N. Le problème de construction d'un réseau de structure déterminée revient à savoir si la matrice structurelle correspondante appartient ou non à $\mathfrak{M}_N$.

Un problème important du point de vue pratique est celui de la recherche d'une matrice ayant le plus petit nombre possible d'unités (notée $M_{N_{\min}}$): elle détermine une structure à nombre de liaisons minimal. Soient données $\pi_1, \ldots, \pi_n, \tau_1, \ldots, \tau_n$. La solution est triviale:

1. En première itération on teste les partitions de l'ensemble $\{\pi_1, \ldots, \pi_n\}$ cherchant celles des τ_j qui vérifient l'inégalité $\pi_i \leqslant \tau_j$, τ_j étant une partition de l'ensemble $\{\tau_1, \ldots, \tau_n\}$. Dans le cas $\pi_i \leqslant \tau_j$, on pose $m_{ij} = 1$ et la colonne τ_j est supprimée de la matrice.

2. On fait tous les produits possibles deux à deux $\pi_i\pi_k$ ($i, k = 1, \ldots, n$; $i \neq k$) et on cherche des τ_p telles que $\pi_i\pi_k \leqslant \tau_p$. Ici τ_p désigne une partition parmi celles qui sont restées dans la matrice après la première itération. On admet que $m_{ip} = m_{kp} = 1$, et la colonne τ_p est supprimée de la matrice.

3. Ceci fait, on opère par itérations successives pour tous les produits trois à trois $\pi_i\pi_k\pi_r$, puis quatre à quatre, et ainsi de suite, jusqu'à ce que toutes les colonnes de la matrice susceptibles d'être supprimées le soient.

La matrice structurelle permet de dégager du théorème de décomposition général les conditions nécessaires et suffisantes pour la construction d'un réseau dont la structure est soumise à telle ou telle contrainte.

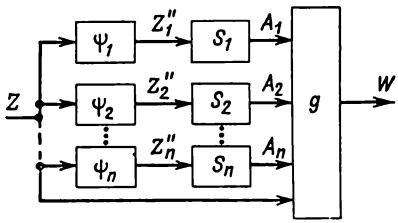


Fig. 4-15. Décomposition parallèle

Décomposition parallèle. Dans ce cas (fig. 4-15) tous les $Z_i' = \varnothing$, $i = 1, \ldots, n$, ce qui veut dire que f_i ne reçoit aucun signal provenant d'un autre automate composant. Il est clair que la matrice structurelle correspondante ne peut avoir des unités que dans sa diagonale principale, c'est-à-dire que pour $i \neq j$ on a

$$m_{ij} = 0, \quad i, j = 1, \ldots, n. \tag{4-4}$$

Puisque dans la colonne τ_i l'unité ne peut figurer que dans la ligne π_i, on a $\pi_i \leqslant \tau_i$, $i = 1, \ldots, n$. Il est montré dans [28] qu'une partition pareille possède la propriété de substitution (p.s.). Ainsi donc, pour qu'il existe une décomposition parallèle à n automates composants, il faut et il suffit que l'ensemble d'états de l'automate à décomposer admette n partitions p.s.

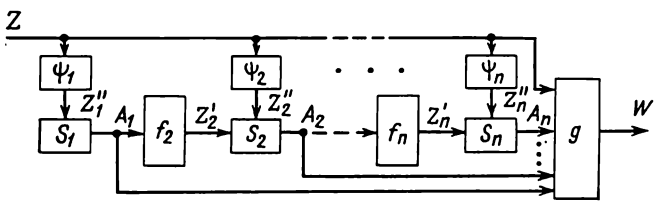


Fig. 4-16. Décomposition série

Décomposition série. Dans le cas général (fig. 4-16) le réseau admet la matrice structurelle montrée dans le tableau 4-58. La condition d'existence d'une décomposition série s'énonce évidemment comme suit:

$$\pi_1 \leqslant \tau_1, \quad \pi_{i-1}\pi_i \leqslant \tau_i, \quad 2 \leqslant i \leqslant n,$$

Tableau 4-58

Matrice structurelle de la décomposition série

	τ_1	τ_2	τ_3	...	τ_{n-2}	τ_{n-1}	τ_n
π_1	1	1					
π_2		1	1				
π_3			1				
π_{n-2}					1	1	
π_{n-1}						1	1
π_n							1

autrement dit, il doit exister une partition p.s. définissant le premier automate composant. Dans le cas où l'information externe ne parvient qu'à S_1, on a

$$\eta_1 \neq \eta\ (1),\ \eta_2 = \eta_3 = \ldots = \eta_n = \eta\ (1).$$

CHAPITRE 5

AUTOMATE STRUCTURÉ

5-1. Méthode canonique de synthèse structurelle des automates

L'étape de synthèse abstraite, qui se termine par la minimisation du nombre d'états de l'automate, est suivie de la synthèse structurelle, c'est-à-dire de la construction d'un schéma réalisant l'automate avec des éléments logiques du type donné. L'automate abstrait n'est qu'un modèle mathématique de circuit séquentiel ; par contre, l'*automate structuré* tient compte déjà de la nature des signaux d'entrée et de sortie, ainsi que de la structure interne de l'automate décrit par un schéma de structure. L'objectif fondamental de la *théorie structurelle des automates* consiste dans la recherche des méthodes générales de composition de schémas de structure des automates à partir d'automates élémentaires choisis parmi un nombre fini de types, défini à l'avance.

A la différence d'un C-automate abstrait possédant une voie d'entrée et deux voies de sortie réservées aux signaux appartenant à l'alphabet d'entrée $Z = \{z_1, \ldots, z_f, \ldots, z_F\}$ et aux alphabets de sortie $W = \{w_1, \ldots, w_g, \ldots, w_G\}$ et $U = \{u_1, \ldots, u_h, \ldots \ldots, u_H\}$ de l'automate, un automate structuré présente des voies d'entrée $x_1, \ldots, x_l, \ldots, x_L$ et de sortie $y_1, \ldots, y_n, \ldots, y_N$, $r_1, \ldots, r_d, \ldots, r_D$ réservées aux signaux qui proviennent de l'alphabet structuré de l'automate.

L'ensemble des valeurs que peuvent prendre les signaux attaquant chaque unité d'entrée (de sortie) constitue l'*alphabet structuré d'entrée* (*de sortie*) de l'automate. A l'heure actuelle, l'alphabet structuré le plus employé est l'alphabet binaire, en raison de la facilité avec laquelle il se réalise dans les composants et les appareils courants. En outre, la théorie bien développée des fonctions booléennes milite en faveur du système binaire, car de nombreuses opérations sur les circuits peuvent se faire de façon formelle. Ce sont les raisons pour lesquelles, dans le texte qui suit, les problèmes de synthèse structurelle des automates seront traités principalement en alphabet structuré binaire.

Une fois qu'on a adopté l'alphabet binaire, chaque signal d'entrée z_f et de sortie w_g, u_h de l'automate abstrait se laisse coder par un

vecteur de longueur L, N, D respectivement; les composantes du vecteur ne peuvent prendre que deux valeurs: 0 ou 1:

$$z_f \to (e_{f1}, \ldots, e_{fl}, \ldots, e_{fL}),\ e_{fl} \in \{0, 1\},\ f = 1, \ldots, F;$$
$$w_g \to (e_{g1}, \ldots, e_{gn}, \ldots, e_{gN}),\ e_{gn} \in \{0, 1\},\ g = 1, \ldots, G;$$
$$u_h \to (e_{h1}, \ldots, e_{hd}, \ldots, e_{hD}),\ e_{hd} \in \{0, 1\}, h = 1, \ldots, H.$$

Il est évident que $L \geqslant]\log_2 F[$, $N \geqslant]\log_2 G[$, $D \geqslant]\log_2 H[$, où $]a[$ est le nombre entier le plus voisin supérieur à a, ou égal à a si a est déjà un nombre entier.

L'étape de synthèse structurelle commence par le choix des automates élémentaires. Puis, en composant ces derniers, on construit le schéma de structure de l'automate de Mealy ou de Moore ou du C-automate obtenu à l'étape de synthèse abstraite. Si le problème de synthèse structurelle admet une solution, on dit que le système d'automates demandé est *structurellement complet.*

La méthode canonique de synthèse structurelle des automates repose sur le théorème définissant la notion de système structurellement complet, démontré dans [22]: *Tout système d'automates élémentaires qui contient un automate de Moore à mémoire non triviale, pourvu d'un système complet de transitions et d'un système complet de sorties, et un système fonctionnellement complet d'éléments logiques est structurellement complet.*

Il existe une méthode de construction générale (*méthode canonique*) par laquelle on arrive, dans le cas considéré, à réduire le problème de synthèse structurelle d'un automate arbitraire à un problème de synthèse d'un circuit combinationnel.

La méthode canonique fournit un système d'équations logiques qui définissent les signaux de sortie de l'automate et les signaux appliqués aux entrées des éléments de mémoire en fonction des signaux attaquant l'entrée externe de l'automate et des signaux prélevés sur la sortie des éléments de mémoire. Les équations indiquées sont dites *équations canoniques.*

Sous peine de compromettre le fonctionnement correct du circuit, on doit évidemment empêcher que les signaux attaquant les entrées des éléments de mémoire participent directement à la formation des signaux de sortie qui, à ce même instant, sont rebouclés sur les mêmes entrées. Il s'ensuit que les éléments de mémoire doivent être constitués non par des automates de Mealy mais par des automates de Moore (leurs équations de fonctionnement sont données dans le § 2-2).

Ainsi donc, tout système structurellement complet d'automates élémentaires doit nécessairement contenir au moins un automate de Moore. D'autre part, si l'on veut minimiser le nombre d'éléments de mémoire, on choisira comme éléments de mémoire des automates

de Moore dits *complets*, c'est-à-dire pourvus d'un système complet de transitions et d'un système complet de sorties.

Afin d'illustrer la notion d'automate de mémoire complet, considérons un automate de Moore M_i dans lequel $Q = \{q_1, q_2, q_3\}$, $W = \{w_1, w_2, w_3\}$, $B = \{b_1, b_2, b_3\}$ sont respectivement les alphabets d'entrée, de sortie et des états internes; les fonctions des transitions et des sorties de M_i sont définies par une table des transitions marquée (tableau 5-1).

Tableau 5-1

$\delta : B \times Q \rightarrow B$
$\lambda : B \rightarrow W$

	w_2	w_1	w_3
	b_1	b_2	b_3
q_1	b_1	b_2	b_3
q_2	b_2	b_3	b_1
q_3	b_3	b_1	b_2

En disant que le *système des transitions est complet*, on sous-entend que pour tout couple d'états (b_m, b_s) de l'automate il existe un signal d'entrée qui fait passer du premier élément du couple (b_m) au second (b_s); autrement dit, chaque colonne de la table des transitions d'un tel automate doit contenir tous ses états.

En disant que le système des sorties (de l'automate de Moore, voir § 4-1) est complet, on sous-entend que chaque état de l'automate se fait correspondre un signal de sortie particulier, distinct des signaux de sortie qui correspondent aux autres états. Dans un tel automate, le nombre des signaux de sortie est nécessairement égal au nombre des états. Compte tenu de ce qui précède, on peut dire que dans un automate de Moore à système complet de sorties, les états peuvent être identifiés aux signaux de sortie. Nous utiliserons donc, dans les automates de mémoire, les mêmes notations pour les états que pour les signaux de sortie, ce qui fait que la table des transitions marquée de l'automate de Moore à système complet de sorties devient une simple table des transitions. L'automate de Moore défini dans le tableau 5-1 possède des systèmes complets de transitions et de sorties: son alphabet de sortie pourra donc être $B = \{b_1, b_2, b_3\}$.

Selon la méthode canonique de synthèse structurelle, le schéma de structure d'un C-automate se présente en trois parties: une mémoire et deux circuits combinationnels CC1 et CC2 (fig. 5-1). Expliquons la destination de chaque partie du schéma.

La *mémoire de l'automate* est constituée par des automates de mémoire préalablement choisis: ce sont des automates de Moore élémentaires complets $M_1, \ldots, M_r, \ldots, M_R$. Une fois les éléments de mémoire choisis, chaque état a_m $(m = 1, \ldots, M)$ du C-automate abstrait est codé dans l'automate structuré par un vecteur $(e_{m1}, \ldots, e_{mr}, \ldots, e_{mR})$ qui a comme composantes les états des éléments de mémoire M_r $(r = 1, \ldots, R)$. Si tous les automates

$M_1, \ldots, M_R$ sont identiques (ce qui n'a pas toujours lieu dans le cas général), leur nombre est égal à

$$R \geqslant \lceil \log_\theta M \rceil,$$

où θ est le nombre des états d'un automate de mémoire élémentaire.

Supposons que le C-automate abstrait passe de l'état a_m à l'état a_s sous l'action du signal d'entrée z_f en produisant le signal de sortie

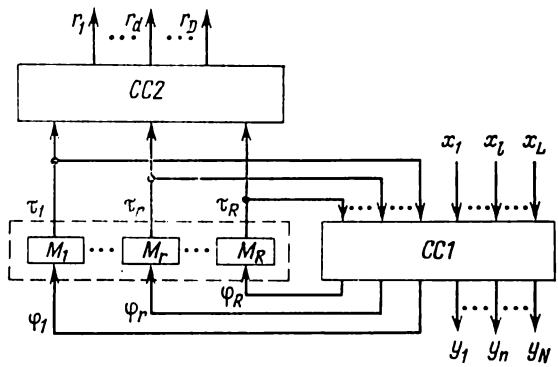


Fig. 5-1. Un C-automate composé d'une mémoire et de deux circuits combinationnels.

w_g. Alors l'automate structuré correspondant passe d'un état $(e_{m1}, \ldots, e_{mR})$ à un état $(e_{s1}, \ldots, e_{sR})$ sous l'action d'un signal d'entrée $(e_{f1}, \ldots, e_{fL})$ en produisant un signal de sortie $(e_{g1}, \ldots, e_{gN})$ du type 1. Dans une telle transition, les éléments de mémoire changent d'état sous l'action des signaux $\varphi = (\varphi_1, \ldots, \varphi_r, \ldots, \varphi_R)$ qui arrivent sur les entrées de la mémoire en provenance de la sortie du CC1. Une fois passé à l'état a_s (ce qui correspond à un état $(e_{s1}, \ldots, e_{sR})$ dans l'automate structuré), l'automate produit le signal de sortie u_h du type 2 (resp. $(e_{h1}, \ldots, e_{hD})$) pendant toute la durée de séjour en cet état.

Le circuit combinationnel CC1 forme les signaux de sortie du type 1 et les signaux d'entrée des automates de mémoire. Le circuit combinationnel CC2 forme les signaux du type 2.

On a pris dans cet exemple un C-automate, car ce modèle généralise les automates de Mealy et de Moore. S'il s'agit de faire la synthèse d'un automate de Mealy, on admet que le C-automate n'a pas de fonction λ_2 et l'on supprime CC2. Pour un automate de Moore, c'est la fonction λ_1 qui est inexistante; le circuit CC1 n'a pas alors de voies de sortie $y_1, \ldots, y_N$.

Ainsi donc, après avoir choisi les éléments de mémoire et codé les états, la synthèse de l'automate structuré se réduit à celle d'un

circuit combinationnel susceptible de réaliser les fonctions

$$
\begin{gathered}
y_1 = y_1(\tau_1, \ldots, \tau_R, x_1, \ldots, x_L), \\
\cdots\cdots\cdots\cdots\cdots\cdots \\
y_N = y_N(\tau_1, \ldots, \tau_R, x_1, \ldots, x_L), \\
\varphi_1 = \varphi_1(\tau_1, \ldots, \tau_R, x_1, \ldots, x_L), \\
\cdots\cdots\cdots\cdots\cdots\cdots \\
\varphi_R = \varphi_R(\tau_1, \ldots, \tau_R, x_1, \ldots, x_L), \\
r_1 = r_1(\tau_1, \ldots, \tau_R), \\
\cdots\cdots\cdots\cdots \\
r_D = r_D(\tau_1, \ldots, \tau_R),
\end{gathered}
$$

où $\tau = (\tau_1, \ldots, \tau_R)$ est la *fonction de rétroaction*, ou *de retour*, entre la mémoire de l'automate et le circuit combinationnel; la fonction $\varphi(\varphi_1, \ldots, \varphi_R)$ est appelée *fonction d'excitation de la mémoire* de l'automate.

Un automate de mémoire peut être considéré, lui aussi, à un niveau abstrait et à un niveau structurel. L'automate de mémoire

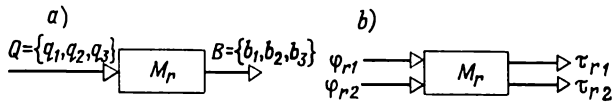


Fig. 5-2. Automate de mémoire: abstrait (*a*), structuré (*b*)

abstrait M_r défini dans le tableau 5-1 *) possède une voie d'entrée et deux voies de sortie (fig. 5-2, *a*).

Pour passer de l'automate abstrait à l'automate structuré, il est nécessaire de coder ses signaux d'entrée et de sortie par des collections de signaux dans les voies d'entrée et de sortie. Si l'on utilise l'alphabet structuré binaire, l'automate M_r possède deux voies d'entrée et deux voies de sortie (fig. 5-2, *b*). Dans le cas général où l'automate de mémoire abstrait admet h états ($B = \{b_1, \ldots, b_h\}$) et accepte p signaux d'entrée ($Q = \{q_1, \ldots, q_p\}$), le nombre de ses voies d'entrée K et le nombre de ses voies de sortie T vérifient les inégalités respectives suivantes:

$$K \geqslant \log_\eta p, \qquad T \geqslant \log_\nu h,$$

où η et ν sont les nombres des lettres des alphabets structurés d'entrée et de sortie respectivement de l'automate de mémoire M_r. Re-

*) Après l'identification des alphabets de sortie et des états internes, l'automate abstrait M_r admet comme signaux de sortie $\{b_1, b_2, b_3\}$.

venant à la figure 5-1, on doit noter que les composantes φ_r et τ_r ($r = 1, \ldots, R$) des vecteurs des signaux d'excitation de la mémoire φ et des signaux de retour τ peuvent être représentées à leur tour par des vecteurs $\varphi_r = (\varphi_{r1}, \ldots, \varphi_{rK})$, $\tau_r = (\tau_{r1}, \ldots, \tau_{rT})$.

Comme nous l'avons déjà dit, nous utiliserons toujours (à moins d'indication contraire) l'alphabet structuré binaire pour les voies d'entrée et de sortie tant de l'automate cherché que des automates de mémoire. Quant à l'alphabet des états internes des automates de mémoire, il sera dans la plupart des cas binaire lui aussi : cela veut dire que les éléments de mémoire seront des automates à deux états.

En définissant les fonctions d'excitation de mémoire de l'automate, nous ferons intervenir la *fonction des entrées de l'élément de mémoire* μ (b_m, b_s) qui associe à chaque couple d'états (b_m, b_s) un signal qui, appliqué à l'entrée de l'automate en question, provoque la transition de b_m à b_s. Cette fonction se prête facilement à une définition tabulaire. Pour l'automate de mémoire dont la fonction des transitions est donnée dans le tableau 5-1, la fonction des entrées est définie par le tableau 5-2, dans lequel le signal d'entrée μ (b_m, b_s) figure dans la deuxième colonne, encadré par b_m et b_s. En examinant ce tableau, on voit que la transition de b_1 à b_2 est induite par q_2 ; de b_3 à b_3, par q_1, etc. Si les signaux d'entrée et les signaux de sortie (états) de l'automate de mémoire sont codés par des collections de signaux ($\varphi_{r1}, \ldots, \varphi_{rK}$), ($\tau_{r1}, \ldots, \tau_{rT}$) sur ses voies d'entrée et de sortie respectivement, le tableau de la fonction des entrées présente au lieu des signaux $q_1, \ldots, q_p$ des collections

Tableau 5-2

Fonction des entrées de l'automate de mémoire abstrait

b_m	q	b_s
b_1	q_1	b_1
b_1	q_2	b_2
b_1	q_3	b_3
b_2	q_3	b_1
b_2	q_1	b_2
b_2	q_2	b_3
b_3	q_2	b_1
b_3	q_3	b_2
b_3	q_1	b_3

Tableau 5-3

Fonction des entrées de l'automate de mémoire structuré

$\tau_{r_1}\tau_{r_2}$	$\varphi_{r_1}\varphi_{r_2}$	$\tau_{r_1}\tau_{r_2}$
00	00	00
00	01	10
00	10	11
10	10	00
10	00	10
10	01	11
11	01	00
11	10	10
11	00	11

correspondantes de signaux. Ainsi, introduisant pour notre exemple (tableau 5-2) les codes:

$$q_1 \to 00, \quad q_2 \to 01, \quad q_3 \to 10,$$
$$b_1 \to 00, \quad b_2 \to 10, \quad b_3 \to 11,$$

nous aboutirons au tableau 5-3 pour la fonction des entrées. On voit que pour passer par exemple de l'état 00 à l'état 10, on doit appliquer 0 sur la première voie d'entrée et 1 sur la deuxième; pour passer de l'état 11 à l'état 11, on doit envoyer 0 dans chacune des voies d'entrée.

Dans le paragraphe suivant, nous examinerons en détail un exemple d'application de la méthode canonique à la synthèse, au niveau structurel, des automates définis par leurs tables des transitions et des sorties. Notons tout de suite que de pareils cas ne se présentent presque jamais dans la pratique de conception des circuits séquentiels; or, pour bien assimiler la méthodologie de synthèse des automates microprogrammés compliqués décrite dans les chapitres qui vont suivre, le lecteur doit bien comprendre les procédures exposées dans ce chapitre.

5-2. Exemple d'application de la méthode canonique de synthèse structurelle

On demande de composer le C-automate partiel S défini par la table des transitions (tableau 5-4) et par la table des sorties marquée (tableau 5-5); les éléments de base seront ET, OU, NON et l'automate de mémoire M dont la fonction des transitions est donnée dans le tableau 5-6.

Tableau 5-4

$\delta : A \times Z \to A$

$Z \backslash A$	a_1	a_2	a_3
z_1	a_2	—	a_1
z_2	a_3	a_1	—
z_3	a_2	a_3	a_3

Tableau 5-5

$\lambda_1 : A \times Z \to W, \quad \lambda_2 : A \to U$

U	u_1	u_2	u_1
$Z \backslash A$	a_1	a_2	a_3
z_1	w_3	—	w_2
z_2	w_4	w_3	—
z_3	w_2	w_1	w_3

Tableau 5-6

$B \times Q \to B$

Q \ B	b_1	b_2
q_1	b_1	b_2
q_2	b_2	b_1

Tableau 5-7

$Q \to \varphi$

Q	φ
q_1	0
q_2	1

Tableau 5-8

$B \to \tau$

B	τ
b_1	0
b_2	1

Nous convenons que les alphabets structurés d'entrée, de sortie et des états internes de l'automate S et de l'automate de mémoire sont binaires.

Pour donner une représentation structurelle à S et M, on doit coder leurs signaux d'entrée et de sortie et leurs états.

Puisqu'au niveau abstrait l'automate M admet deux signaux d'entrée et deux signaux de sortie, au niveau structuré il aura une voie d'entrée (φ) et une voie de sortie (τ). Après le codage des si-

Tableau 5-9

$\tau \times \varphi \to \tau$

φ \ τ	0	1
0	0	1
1	1	0

Tableau 5-10

Fonction des entrées de l'automate M structuré

$\tau_{\text{init.}}$	φ	$\tau_{\text{trans.}}$
0	0	0
0	1	1
1	1	0
1	0	1

Tableau 5-11

Codage des états de l'automate S

A	τ_1	τ_2
a_1	0	0
a_2	0	1
a_3	1	1

Tableau 5-12

Codage des signaux d'entrée de l'automate S

Z	x_1	x_2
z_1	0	0
z_2	0	1
z_3	1	0

Tableau 5-13

Codage des signaux de sortie du type 1 de l'automate S

W	y_1	y_2
w_1	1	0
w_2	0	0
w_3	1	1
w_4	0	1

Tableau 5-14

Codage des signaux de sortie du type 2 de l'automate S

U	r
u_1	1
u_2	0

gnaux d'entrée (tableau 5-7) et des états (tableau 5-8) de l'automate M abstrait, sa table des transitions prend la forme du tableau 5-9.

Tableau 5-15

Table des transitions de l'automate structuré *S*

	00	01	11
00	01	—	00
01	11	00	—
10	01	11	11

Tableau 5-16

Table des sorties repérée de l'automate structuré *S*

	1	0	1
	00	01	11
00	11	—	00
01	01	11	—
10	00	10	11

La fonction des entrées de l'automate M structuré est donnée dans le tableau 5-10.

Puisque l'automate S abstrait admet trois états, l'automate S structuré devra posséder deux éléments de mémoire M_1, M_2. Trois signaux d'entrée abstraits (z_1, z_2, z_3) et quatre signaux de sortie du type 1 (w_1, w_2, w_3, w_4) nécessitent deux voies d'entrée (x_1, x_2) et deux voies de sortie (y_1, y_2). Pour la réalisation des deux signaux de sortie du type 2 (u_1, u_2), on a besoin d'une autre voie de sortie (r). Les résultats du codage des états et des signaux d'entrée et de sortie de l'automate S sont résumés dans les tableaux 5-11 à 5-14.

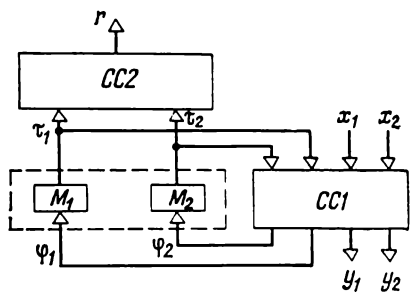


Fig. 5-3. Schéma de structure de l'automate S

Remplaçant dans les tableaux 5-4, 5-5 les états et les signaux correspondants par leurs codes, nous obtenons la table des transitions (tableau 5-15) et la table des sorties marquée (tableau 5-16) de l'automate structuré S dont le schéma-block est donné sur la figure 5-3. Ainsi donc, les éléments de mémoire choisis et les états codés, la synthèse du C-automate se réduit à celle de deux circuits combinationnels CC1, CC2 qui réalisent les fonctions

$$y_1(\tau_1, \tau_2, x_1, x_2), \quad y_2(\tau_1, \tau_2, x_1, x_2),$$
$$\varphi_1(\tau_1, \tau_2, x_1, x_2), \quad \varphi_2(\tau_1, \tau_2, x_1, x_2),$$
$$r(\tau_1, \tau_2).$$

Rappelons que dans un automate abstrait le signal de sortie à l'instant t dépend de l'état interne de l'automate et du signal d'entrée au même instant t, de sorte que $w(t) = \lambda_1(a(t), z(t))$, $u(t) = \lambda_2(a(t))$. L'automate structuré conserve bien sûr ces relations temporelles, si bien qu'on a $y_i = y_i(\tau_1(t), \tau_2(t), x_1(t), x_2(t))$, $i = 1, 2$; $r(t) = r(\tau_1(t), \tau_2(t))$. Le tableau 5-16 résume donc, sous une forme condensée, les fonctions y_1, y_2 de τ_1, τ_2, x_1, x_2 et la fonction r de τ_1, τ_2. Par analogie à la définition tabulaire des fonctions booléennes, les expressions analytiques de y_1, y_2 se déduisent immédiatement du tableau 5-16 : ce sont les disjonctions des conjonctions correspondant aux combinaisons de variables τ_1, τ_2, x_1, x_2 pour lesquelles ces fonctions prennent la valeur 1. Il ressort du tableau 5-16 que

$$\left.\begin{aligned} y_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2, \\ y_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \tau_1\tau_2x_1\bar{x}_2. \end{aligned}\right\} \qquad (5\text{-}1)$$

Identifions chaque combinaison binaire de valeurs de variables (τ_1, τ_2, x_1, x_2) à son numéro (en écrivant par exemple 1 au lieu de 0001 et 12 au lieu de 1100) : les expressions (5-1) s'écriront alors comme suit :

$$y_1 = 0 \vee 5 \vee 6 \vee 14, \quad y_2 = 0 \vee 1 \vee 5 \vee 14. \qquad (5\text{-}2)$$

L'expression de la fonction $r = r(\tau_1, \tau_2)$ se déduit comme les précédentes directement du tableau 5-16 :

$$r = \bar{\tau}_1\bar{\tau}_2 \vee \tau_1\tau_2. \qquad (5\text{-}3)$$

Par contre, les expressions des fonctions d'excitation de la mémoire ne s'obtiennent pas si facilement. Voyons par exemple ce que devient l'automate S qui, se trouvant dans l'état 01, se voit attaqué par le signal d'entrée 10. Le tableau 5-15 montre (deuxième colonne, troisième ligne) que l'automate passe de l'état 01 à l'état 11. Cette transition se compose de deux transitions des automates de mémoire élémentaires. Le premier automate, M_1, passe de l'état 0 à l'état 1, et le deuxième, M_2, qui était dans 1, conserve cet état.

Tableau 5-17

Fonction d'excitation de la mémoire de l'automate S

	00	01	11
00	01	—	11
01	11	01	—
10	01	10	00

Les transitions des automates de mémoire sont induites par des signaux d'excitation appliqués sur leurs entrées. Pour savoir quel signal on doit appliquer sur l'entrée de l'automate M_1 pour provoquer la transition de 0 à 1, regardons le tableau 5-10 qui définit la fonction des entrées de l'automate de mémoire. On trouve dans la deu-

xième ligne le signal 1. D'une façon analogue, on constate que pour l'automate M_2, la transition de 1 à 1 est conditionnée par le signal 0 (quatrième ligne). Ainsi donc, si l'on veut que l'automate S passe de l'état 01 à l'état 11, on doit envoyer dans les entrées de sa mémoire le signal d'excitation vectoriel 10. Inscrivons ce résultat à sa place dans le tableau de la fonction d'excitation (tableau 5-17), à l'intersection de la deuxième colonne et de la troisième ligne. Puisque la fonction d'excitation de la mémoire de l'automate dépend des mêmes variables τ_1, τ_2, x_1, x_2, la notation des colonnes et des lignes du tableau 5-17 est la même que dans le tableau 5-15.

Ensuite, en prenant toutes les autres transitions résumées dans le tableau 5-15, on compose le tableau complet de la fonction d'excitation de la mémoire de l'automate S. Les expressions pour les valeurs unité des fonctions φ_1 et φ_2 se déduisent directement du tableau 5-17 :

$$\left.\begin{aligned} \varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 1 \vee 6 \vee 12, \\ \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \\ &\qquad \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 0 \vee 1 \vee 2 \vee 5 \vee 12. \end{aligned}\right\} \qquad (5\text{-}4)$$

Sans entrer dans le détail de la minimisation du circuit combinationnel de S, nous construirons le schéma logique de S directement à partir des expressions 5-1, 5-3 et 5-4 (fig. 5-4).

Remarquons que, pour minimiser les expressions établies, on pourrait utiliser les combinaisons binaires de variables pour lesquelles les fonctions correspondantes sont non définies. Formulons le critère d'indétermination :

1. Si un code $(e_{m1}, \ldots, e_{mr}, \ldots, e_{mR})$, $e_{mr} \in \{0, 1\}$, n'est pas employé pour le codage des états de l'automate, toutes les composantes des fonctions d'excitation de la mémoire et des fonctions des sorties (du type 1) ne sont pas définies pour toutes les combinaisons $(e_{m1}, \ldots, e_{mr}, \ldots, e_{mR}, e_{f1}, \ldots, e_{fl}, \ldots, e_{fL})$, $e_{fl} \in \{0, 1\}$, obtenues à partir de ce code et de tous les codes des signaux d'entrée. Dans notre exemple le code 10 n'est pas employé pour le codage des états, aussi les combinaisons

$$\begin{pmatrix} 1\,0\,0\,0 \\ 1\,0\,0\,1 \\ 1\,0\,1\,0 \\ 1\,0\,1\,1 \end{pmatrix}$$

appartiennent au domaine d'indétermination des fonctions φ_1, φ_2, y_1, y_2. D'une façon parfaitement analogue, la fonction r est non définie pour $\tau_1 = 1$, $\tau_2 = 0$.

2. Si un code $(e_{f1}, \ldots, e_{fl}, \ldots, e_{fL})$, $e_{fl} \in \{0, 1\}$, n'est pas employé pour le codage des signaux d'entrée, toutes les composantes des fonctions d'excitation de la mémoire et des fonctions des sorties (du type 1) ne sont pas définies pour toutes les combinaisons $(e_{m1}, \ldots, e_{mr}, \ldots, e_{mR}, e_{f1}, \ldots, e_{fl}, \ldots, e_{fL})$, $e_{mr} \in \{0, 1\}$, obtenues

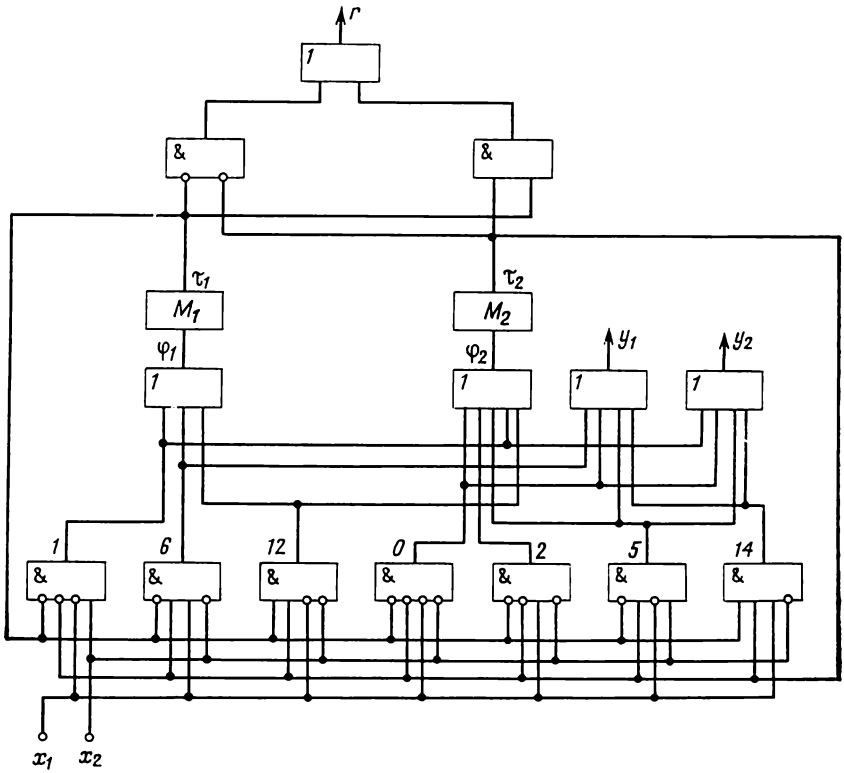


Fig. 5-4. Schéma logique de l'automate S

à partir de ce code et de tous les codes des états. Dans notre exemple le code 11 n'est pas employé pour le codage des signaux d'entrée, aussi les combinaisons

$$\begin{pmatrix} 0\,0\,1\,1 \\ 0\,1\,1\,1 \\ 1\,0\,1\,1 \\ 1\,1\,1\,1 \end{pmatrix}$$

appartiennent au domaine d'indétermination des fonctions φ_1, φ_2, y_1, y_2.

3. Si la fonction des transitions est non définie sur un couple (a_m, z_f), toutes les composantes de la fonction d'excitation sont non définies pour la combinaison $(e_{m1}, \ldots, e_{mr}, \ldots, e_{mR}, e_{f1}, \ldots \ldots, e_{fl}, \ldots, e_{fL})$, $e_{fl} \in \{0, 1\}$. Ainsi donc, les tirets dans la table d'excitation correspondent aux tirets de la table des transitions. Directement du tableau 5-17 on tire deux combinaisons de ce type :

$$\begin{pmatrix} 0\,1\,0\,0 \\ 1\,1\,0\,1 \end{pmatrix}.$$

4. Par analogie au n° 3 ci-dessus, si la fonction des sorties (du type 1) est non définie sur un couple (a_m, z_f), toutes les composantes de la fonction des sorties (du type 1) sont non définies pour la combinaison qui correspond à ce couple. Du tableau 5-16 on tire deux combinaisons :

$$\begin{pmatrix} 0\,1\,0\,0 \\ 1\,1\,0\,1 \end{pmatrix}.$$

5-3. Synthèse de l'automate avec des bascules D, T et RS

La recherche des fonctions d'excitation de la mémoire de l'automate peut être sensiblement simplifiée pour certains éléments de mémoire usités. Nous examinerons ici la synthèse de l'automate avec des bascules des types D, T et RS. Les méthodes proposées seront illustrées sur l'exemple de l'automate S défini dans les tableaux 5-15 et 5-16. Les fonctions des sorties seront construites dans tous les cas d'après les expressions (5-1) et (5-3), car la table des sorties de S ne dépend pas du choix de tel ou tel élément de mémoire.

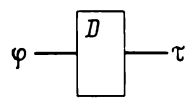


Fig. 5-5. Bascule D

La *bascule du type D* (*élément de retard*) présente une entrée et une sortie et réalise le retard du signal appliqué sur son entrée pendant une durée égale à la période d'horloge (fig. 5-5). La fonction des transitions et la fonction des entrées de cet automate sont résumées dans les tableaux 5-18 et 5-19 respectivement.

Tableau 5-18

Fonction des transitions de la bascule D

	0	1
0	0	0
1	1	1

Tableau 5-19

Fonction des entrées de la bascule D

$\tau_{\text{init.}}$	φ	$\tau_{\text{trans.}}$
0	0	0
0	1	1
1	0	0
1	1	1

En examinant le tableau 5-19, on voit que l'état de la bascule D consécutif à la transition (colonne de droite) est identique au signal d'entrée qui a induit cette transition (colonne de milieu). Pour cette raison la table des fonctions d'excitation de la mémoire de l'automate S coïncide très exactement avec sa table des transitions (tableau

5-15). En effet, les colonnes et les lignes de ces tables portent les mêmes désignations; à l'intersection de l'i-ième ligne et de la j-ième colonne de la table des transitions, on trouve le code de l'état que prend l'automate se trouvant dans l'état associé à la j-ième colonne et attaqué par le signal d'entrée correspondant à la i-ième ligne. Or, avec une bascule D, le code de l'état consécutif à la transition se confond avec les signaux appliqués sur ses entrées. On peut donc déduire immédiatement du tableau 5-15 les expressions des fonctions d'excitation de la mémoire d'un automate construit avec des bascules D :

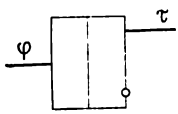


Fig. 5-6. Bascule T

$$\left.\begin{aligned}\varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2 = 1 \vee 6 \vee 14,\\ \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee\\ &\quad \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2 = 0 \vee 1 \vee 2 \vee 6 \vee 14.\end{aligned}\right\} \qquad (5\text{-}5)$$

La *bascule du type T* (*compteur élémentaire, bascule à commande symétrique*) possède une entrée et une sortie (fig. 5-6). Ses fonctions des transitions et des entrées sont résumées dans les tableaux 5-20 et 5-21.

Tableau 5-20

Fonction des transitions de la bascule T

	0	1
0	0	1
1	1	0

Tableau 5-21

Fonction des entrées de la bascule T

$\tau_{init.}$	φ	$\tau_{trans.}$
0	0	0
0	1	1
1	1	0
1	0	1

En regardant le dernier tableau, on remarque que le signal d'entrée figurant dans la colonne de milieu est égal à la somme modulo 2 du code de l'état initial (colonne de gauche) et du code de l'état consécutif à la transition (colonne de droite). Pour un automate construit avec des bascules T, la table des fonctions d'excitation de la mémoire se déduit donc de la table des transitions de la façon suivante.

L'élément de la table des fonctions d'excitation se trouvant à l'intersection de l'i-ième ligne et de la j-ième colonne est égal à la somme modulo 2, digit par digit, du vecteur du code de l'état associé à la j-ième colonne et du vecteur du code de l'état figurant dans la table des transitions à l'intersection de l'i-ième ligne et de la j-ième colonne; autrement dit, l'élément cherché est égal à la somme modulo 2, digit par digit, des codes des états avant la transition et après. La fonction d'excitation obtenue de cette manière pour

l'automate S est donnée dans le tableau 5-22. D'où l'on tire

$$\left.\begin{aligned}\varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 1 \vee 6 \vee 12,\\ \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2x_1x_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee\\ &\qquad \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 0 \vee 1 \vee 2 \vee 5 \vee 12.\end{aligned}\right\} \quad (5\text{-}6)$$

Tableau 5-22

Fonction d'excitation de la mémoire de l'automate S construit avec des bascules T

	00	01	11
00	01	—	11
01	11	01	—
10	01	10	00

Tableau 5-23

Table des transitions de la bascule *RS*

	0	1
00	0	1
01	1	1
10	0	0

La *bascule du type RS* (*bascule à commande asymétrique*, *registre élémentaire*) est représentée sur la figure 5-7 ; sa table des transitions est donnée dans le tableau 5-23. Puisque la bascule RS possède deux entrées, R et S, elle admet quatre combinaisons différentes des signaux d'entrée. Une des combinaisons étant interdite (11), le tableau 5-23 n'a que trois lignes. Les entrées R et S sont respectivement l'entrée 0 et l'entrée 1.

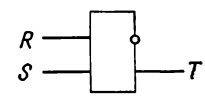


Fig. 5-7. Bascule *RS*

La fonction des entrées de la bascule RS est donnée dans le tableau 5-24. On remarque que si l'on applique la valeur 1 sur l'entrée 0, la bascule passe à l'état 0 quel que soit son état initial ; si l'on applique 1 sur l'entrée 1, la bascule passe à l'état 1.

Tableau 5-24

Fonction des entrées de la bascule *RS*

$T_{\text{init.}}$	RS	$T_{\text{trans.}}$
0	00∨10	0
0	01	1
1	10	0
1	00∨01	1

Tableau 5-25

Fonction des entrées transformée de la bascule *RS*

$T_{\text{init.}}$	RS	$T_{\text{trans.}}$
0	-0	0
0	01	1
1	10	0
1	0-	1

Une autre représentation de la fonction des entrées est proposée dans le tableau 5-25. Ici la transition de 0 à 0 est induite par le signal d'entrée (—0), où le tiret indique que la transition est indépendante du signal appliqué sur l'entrée 0. De même, la transition de 1 à 1 est induite par le signal d'entrée (0-).

Tableau 5-26

Fonction d'excitation de la mémoire de l'automate *S* construit avec des bascules *RS*

	00	01	11
00	-001	—	1010
01	0101	-010	—
10	-001	010-	0-0-

Le tableau 5-25 fournit un système de substitutions utile pour passer de la table des transitions de l'automate structuré à sa table des fonctions d'excitation. L'emploi de ces substitutions sera illustré sur l'exemple de l'automate S (tableau 5-15) dont les fonctions d'excitation sont définies dans le tableau 5-26.

Examinons, dans le tableau 5-15, la transition 00—01 provoquée par le signal d'entrée 00. Cette transition est décomposée en deux transitions partielles dont chacune est réalisée par une bascule. La première bascule passe de 0 à 0 : à cette transition est associée la substitution (—0) (voir tableau 5-25) que nous inscrirons dans le tableau 5-26, à l'intersection de la ligne 00 avec la colonne 00. La seconde partie de cette transition induite par le signal d'entrée 00 est réalisée par la deuxième bascule qui passe de 0 à 1 : la substitution correspondante est (01). Ainsi donc, les signaux d'excitation de la mémoire sont déterminés par la combinaison $R_1S_1R_2S_2 = (-001)$. La transition de S de l'état 00 à l'état 11 induite par le signal d'entrée 01 conduit à la substitution (01) pour la première bascule et à la substitution (01) pour la deuxième, à quoi correspond la combinaison $R_1S_1R_2S_2 = (0101)$ qu'on inscrit dans la deuxième ligne de la première colonne du tableau 5-26. D'une façon analogue, on développe toutes les autres transitions du tableau 5-15, qui donnent la table complète des fonctions d'excitation (tableau 5-26) de l'automate S. Prenant dans ce tableau les combinaisons des variables T_1, T_2, x_1, x_2 qui correspondent à l'application des unités sur les entrées de la première et de la deuxième bascule, on obtient les fonctions d'excitation de la mémoire de l'automate sous forme normale disjonctive :

$$\left.\begin{aligned}
R_1 &= T_1T_2\bar{x}_1\bar{x}_2 = 12,\\
S_1 &= \bar{T}_1\bar{T}_2\bar{x}_1x_2 \vee \bar{T}_1T_2x_1\bar{x}_2 = 1 \vee 6,\\
R_2 &= \bar{T}_1T_2\bar{x}_1x_2 \vee T_1T_2\bar{x}_1\bar{x}_2 = 5 \vee 12,\\
S_2 &= \bar{T}_1\bar{T}_2\bar{x}_1\bar{x}_2 \vee \bar{T}_1\bar{T}_2\bar{x}_1x_2 \vee \bar{T}_1\bar{T}_2x_1\bar{x}_2 = 0 \vee 1 \vee 2.
\end{aligned}\right\} \qquad (5\text{-}7)$$

Il est évident qu'en plus des cas énumérés dans les n°s 1 à 3 du paragraphe précédent, les fonctions d'excitation seront non définies aussi pour chaque combinaison qui correspond à un tiret du tableau 5-26. Par exemple, la fonction R_1 sera non définie pour les combinaisons

$$\begin{pmatrix} 0\,0\,0\,0 \\ 0\,0\,1\,0 \\ 0\,1\,0\,1 \end{pmatrix}.$$

La figure 5-8 présente le schéma logique de l'automate S construit dans la base ET, OU, NON, avec des bascules RS, conformé-

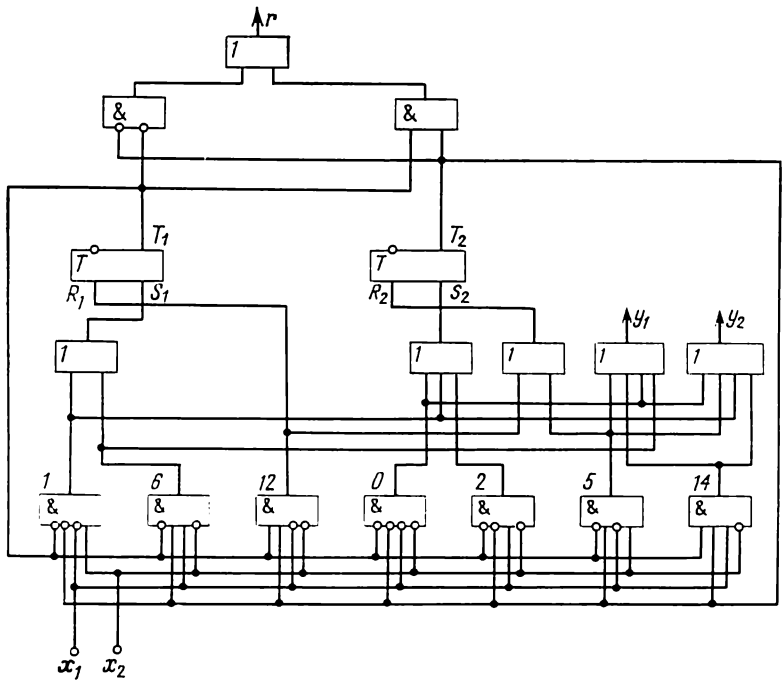


Fig. 5-8. Schéma logique de l'automate S construit avec des bascules RS

ment aux expressions (5-1), (5-3) pour les signaux d'entrée et aux expressions (5-7) pour les fonctions d'excitation de la mémoire.

Les méthodes de synthèse proposées restent valables avec des éléments de mémoire de nature différente. Soit par exemple un auto-

Tableau 5-27

Colonne de la table des transitions d'un automate construit avec des éléments de mémoire de nature différente

0	1	1	0	1	0
1	0	1	0	0	0
0	1	0	1	1	1

Tableau 5-28

Colonne de la table des fonctions d'excitation de la mémoire construite d'après le tableau 5-27

0	1	1	0	1	0
1	0	0	0	10	-0
0	1	1	1	0-	01

mate à deux entrées et possédant six éléments de mémoire, dont deux bascules D (M_1, M_2), deux bascules T (M_3, M_4) et deux bascules RS (M_5, M_6). Une colonne de sa table des transitions est représentée par le tableau 5-27. Alors la colonne correspondante de sa table des fonctions d'excitation aura la forme du tableau 5-28.

Tableau 5-29

Table des transitions de la bascule *JK*

JK	0	1
00	0	1
01	0	0
10	1	1
11	1	0

Tableau 5-30

Fonction des entrées de la bascule *JK*

$T_{init.}$	*JK*	$T_{trans.}$
0	0-	0
0	1-	1
1	-1	0
1	-0	1

Il est clair que des règles analogues, permettant de construire la table des fonctions d'excitation à partir de la table des transitions, peuvent être proposées pour n'importe quels éléments de mémoire. A titre d'exercice, nous proposons au lecteur d'établir ces règles pour la bascule du type JK (fig. 5-9), dont la fonction des transitions et la fonction des entrées sont définies par les tableaux 5-29 et 5-30 respectivement.

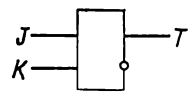


Fig. 5-9. Bascule JK

5-4. Méthode graphique de synthèse structurelle des automates

La méthode graphique de synthèse consiste à représenter un C-automate structuré par un graphe. Comme les termes des expressions du type (5-1) et (5-4) sont des conjonctions des variables d'entrée correspondant aux codes des signaux d'entrée, nous conviendrons de désigner les arcs du graphe de l'automate directement par ces conjonctions, au lieu des signaux d'entrée. C'est ainsi qu'au lieu des signaux 00, 01 et 10 nous marquerons les arcs du graphe par $\bar{x}_1\bar{x}_2$, $\bar{x}_1x_2$ et $x_1\bar{x}_2$ respectivement. En outre, étant donné que, dans les méthodes de synthèse tabulaires, les expressions du type (5-1) sont établies en cherchant dans les colonnes de la table des sorties les composantes des signaux de sortie qui prennent la valeur 1, nous

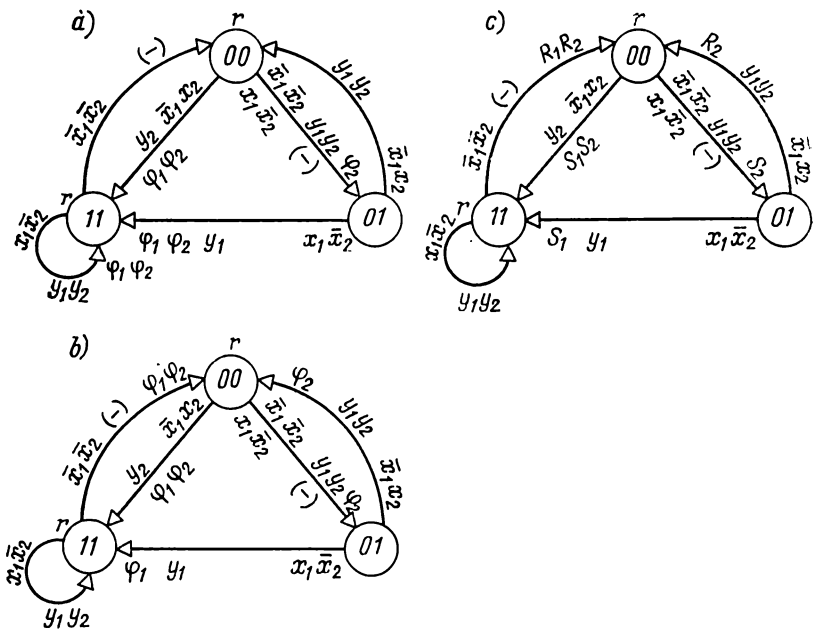


Fig. 5-10. Méthode graphique de synthèse de l'automate S: avec des bascules D (a), avec des bascules T (b), avec des bascules RS (c)

marquerons les arcs du graphe par ces composantes. Par exemple, au lieu des signaux de sortie 10, 01 et 11 nous écrirons y_1, y_2 et y_1y_2 respectivement; au lieu du signal de sortie 00 nous mettrons un tiret (—), car aucune de ses composantes n'est égale à l'unité. De même, pour les signaux de sortie du type 2, nous n'écrirons, à côté de l'état, que la composante du signal de sortie qui prend la valeur 1 dans

l'état considéré. La figure 5-10, *a* montre le graphe de l'automate défini par les tableaux 5-15 et 5-16 (voir plus haut).

La table de la fonction des entrées d'une bascule D (tableau 5-19) montre que si un élément de mémoire M_r prend la valeur 1 pour un état donné, la fonction d'excitation φ_r de cet élément prendra la valeur 1 chaque fois qu'une transition aboutit à cet état. Le tableau 5-21 décrivant la fonction des entrées d'une bascule T permet de voir que la fonction d'excitation φ_r de la bascule M_r prend la valeur 1 à chaque transition qui provoque un changement d'état de cette bascule. La fonction des entrées d'une bascule RS (tableau 5-25) montre qu'une telle bascule passe de 0 à 1 lorsqu'on applique 1 sur son entrée S, et de 1 à 0 en appliquant 1 sur son entrée R. Ces remarques préliminaires permettent de décrire la procédure de synthèse d'un C-automate structuré à partir de son graphe.

Si l'automate se construit avec des bascules du type D, tous les arcs orientés allant à un état où M_r prend la valeur 1 seront marqués par φ_r, $r = 1, \ldots, R$, où R est la longueur des codes d'états. De toute évidence, chaque arc du graphe de l'automate sera affecté d'autant de symboles φ_r qu'il y a d'unités dans le code de l'état auquel l'arc est incident vers l'intérieur (fig. 5-10, *a*).

Si l'automate est construit avec des bascules T, alors chaque fois que la bascule M_r passe de 0 à 1 ou de 1 à 0 lors de la transition correspondante, on marque l'arc de son graphe du symbole φ_r ($r = 1, \ldots, R$). Il est évident qu'on doit porter sur chaque arc autant de symboles qu'il y a d'éléments de mémoire qui changent d'état lors de la transition qui correspond à l'arc donné (fig. 5-10, *b*).

Si l'automate se construit avec des bascules RS, on marque l'arc par S_r si la transition correspondante fait passer la bascule M_r de 0 à 1, et par R_r si M_r passe de 1 à 0 ($r = 1, \ldots, R$). Il est évident que, par analogie au cas des bascules du type T, chaque arc portera l'indication d'autant de symboles qu'il y a d'éléments de mémoire qui changent d'état à la suite de la transition considérée (fig. 5-10, *c*).

Quel que soit le type des éléments de mémoire utilisés, l'expression de φ_r (R_r, S_r), $r = 1, \ldots, R$, a la forme d'une disjonction d'expressions du type $A_m X_r$ dans lesquelles: $A_m = \tau_1^{e_{m1}} \ldots \tau_r^{e_{mr}} \ldots \tau_R^{e_{mR}}$ est la conjonction correspondant au code ($e_{m1}, \ldots, e_{mR}$) de l'état a_m auquel l'arc marqué par φ_r (R_r, S_r) est incident vers l'extérieur; $e_{mr} \in \{0, 1\}$, $\tau_r^0 = \overline{\tau}_r$, $\tau_r^1 = \tau_r$; X_r est le signal d'entrée affecté à l'arc (ou la disjonction des signaux d'entrée s'il y en a plusieurs).

L'expression de y_n ($n = 1, \ldots, N$, où N est le nombre des voies de sortie réservées aux signaux de sortie du type 1) s'obtient sous forme d'une disjonction des expressions du type $A_m X_n$ dans lesquelles: A_m est la conjonction correspondant au code de l'état a_m auquel l'arc marqué par y_n est incident vers l'extérieur; X_n est le

signal d'entrée qui induit la transition provoquant le signal de sortie y_n.

L'expression de r_d ($d = 1, \ldots, D$, où D est le nombre des voies de sortie réservées aux signaux de sortie du type 2) s'obtient sous forme d'une disjonction des conjonctions qui correspondent aux codes des états marqués par r_d.

Ainsi donc, on déduit les expressions pour les fonctions d'excitation de la mémoire et pour les fonctions des sorties directement à partir des graphes montrés sur les figures 5-10, *a*, *b* et *c*. Puisque les signaux de sortie ne dépendent pas du type de l'élément de mémoire, on a pour chacun des trois graphes

$$\left.\begin{aligned} y_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2 = \\ &\qquad = 0 \vee 5 \vee 6 \vee 14, \\ y_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \tau_1\tau_2x_1\bar{x}_2 = \\ &\qquad = 0 \vee 1 \vee 5 \vee 14, \end{aligned}\right\} \tag{5-8}$$

$$r = \bar{\tau}_1\bar{\tau}_2 \vee \tau_1\tau_2. \tag{5-9}$$

Dans le cas de bascules D

$$\left.\begin{aligned} \varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2x_1\bar{x}_2 = 1 \vee 6 \vee 14, \\ \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \\ &\qquad \vee \tau_1\tau_2x_1\bar{x}_2 = 0 \vee 2 \vee 1 \vee 6 \vee 14. \end{aligned}\right\} \tag{5-10}$$

Dans le cas de bascules T

$$\left.\begin{aligned} \varphi_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 1 \vee 6 \vee 12, \\ \varphi_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \\ &\qquad \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 0 \vee 2 \vee 1 \vee 5 \vee 12. \end{aligned}\right\} \tag{5-11}$$

Dans le cas de bascules RS

$$\left.\begin{aligned} R_1 &= \tau_1\tau_2\bar{x}_1\bar{x}_2 = 12, \\ S_1 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 = 1 \vee 6, \\ R_2 &= \bar{\tau}_1\tau_2\bar{x}_1x_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 5 \vee 12, \\ S_2 &= \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 = 1 \vee 0 \vee 2. \end{aligned}\right\} \tag{5-12}$$

Les expressions (5-8), (5-9), (5-10), (5-11) et (5-12) obtenues par la méthode des graphes se confondent respectivement avec les expressions (5-1), (5-3), (5-5), (5-6) et (5-7) obtenues par la méthode tabulaire de synthèse de l'automate S.

5-5. Codage des états. Courses dans l'automate

Comme on l'a déjà souligné plus haut, le problème de codage des états est un des principaux problèmes de la méthode canonique de synthèse structurelle des automates. Remarquons que le codage consiste à introduire une correspondance biunivoque entre l'ensemble $A = \{a_1, \ldots, a_M\}$ des états internes de l'automate et un ensemble de vecteurs à R composantes $\{K_1, \ldots, K_M\}$, $K_m = (e_{m1}, \ldots, e_{mR})$, où e_{mr} est l'état du r-ième élément de mémoire, $r = 1, \ldots, R$. Si $e_{mr} \in \{0, 1\}$, c'est-à-dire si l'alphabet des états des

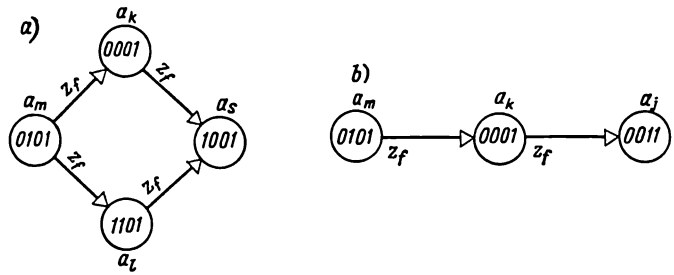


Fig. 5-11. Courses entre les éléments de mémoire: non critiques (aléas) (*a*), critiques (*b*)

éléments de mémoire est binaire, on a $R \geqslant]\log_2 M[$. Pour plus de simplicité, nous nous bornerons à considérer, dans ce paragraphe, des éléments de mémoire constitués par des bascules du type RS; nous les désignerons par $T_1, \ldots, T_R$.

Le changement d'état de l'automate est conditionné par des changements d'état de ses éléments de mémoire. Par exemple, si l'automate passe de l'état a_m, codé 0101, à l'état a_s, codé 1001, cela signifie que sa bascule T_1 passe de 0 à 1, la bascule T_2 de 1 à 0, et les états des bascules T_3 et T_4 restent inchangés.

L'automate peut devenir le siège d'un phénomène appelé *aléas*. Ce phénomène tient à ce que les délais de fonctionnement (de réponse) des différents éléments de mémoire sont différents, bien que suffisamment voisins. En outre, les signaux d'excitation parvenant aux voies d'entrée des automates élémentaires par des chaînes logiques de longueur inégale ont des temps de retards différents. Si un changement d'état de l'automate est déterminé par un changement d'état simultané de plusieurs de ses éléments de mémorisation, il peut y avoir manifestation du phénomène d'aléas: il se peut que l'élément qui change d'état avant les autres ait le temps de modifier, par la chaîne de retour, les signaux devant attaquer les entrées des autres éléments de mémorisation avant que ces derniers passent à l'état suivant. Si ce cas se produit, l'automate peut prendre un état non prévu dans son graphe. Par exemple, l'automate montré sur la figure 5-11, *a* attaqué par le signal d'entrée z_f doit passer normale-

ment de l'état a_m à l'état a_s ; or, avant cette transition définitive, il passe à l'un des deux états transitoires a_k ou a_l, suivant que l'un ou l'autre de ses éléments de mémoire a fonctionné le premier. Si l'automate passe finalement, sous l'action du même signal z_f, à l'état a_s (que ce soit à partir de a_k ou de a_l), on dit que les aléas sont *admissibles*, ou *non critiques*. Par contre, si la structure de l'automate admet une transition de a_k à $a_j \neq a_s$ sous l'action du même signal z_f (fig. 5-11, *b*), l'automate risque de se trouver finalement dans l'état a_j au lieu de a_s, ce qui perturbera son fonctionnement. On a alors des aléas *critiques*, ou *courses*. Lors du codage des états, on doit avoir soin d'éliminer les courses.

Un des moyens d'éliminer les courses consiste à synchroniser les signaux d'entrée de l'automate au moyen des impulsions d'horloge de durée déterminée. On suppose qu'en plus des voies d'entrée $x_1, \ldots, x_L$, l'automate possède une voie d'horloge p qui reçoit à partir d'une horloge le signal $p = 1$ si l'impulsion est présente et $p = 0$ si elle est absente. La transition (a_m, a_s) sera induite dès lors par un signal pz_f, et non par z_f. Alors, si la durée t_p de l'impulsion d'horloge est inférieure à la durée de parcours du plus court chemin dans le circuit combinationnel par le signal de retour synchronisé, on a $k = 0$ au moment où l'automate passe à l'état transitoire a_k (fig. 5-11, *b*) ; on a donc aussi $pz_f = 0$, et les courses deviennent impossibles.

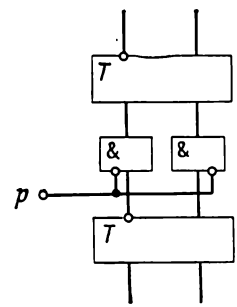


Fig. 5-12. Mémoire doublée

Un autre moyen d'éviter les courses consiste à doubler la mémoire (fig. 5-12). Le transfert d'information de l'élément de mémoire inférieur à l'élément de mémoire supérieur ne peut se faire qu'en l'absence d'impulsion de synchronisation ($p = 0$). Les signaux de retour nécessaires pour l'obtention des fonctions d'excitation et des fonctions des sorties de l'automate sont prélevés sur les bascules de la rangée supérieure. Les courses ne peuvent donc avoir lieu qu'entre les bascules de la rangée inférieure. Les signaux de retour ne peuvent pas changer avant que p devienne nul. Or, à cet instant, le signal d'entrée pz_f devient nul lui aussi, ce qui rend les courses impossibles.

En plus des modifications de structure des automates, les courses peuvent être éliminées par des méthodes de codage spéciales, auxquelles est consacrée une littérature abondante. Par exemple, la méthode de codage décrite dans [42] consiste en ce qui suit.

Soient (α, β) et (γ, δ) deux couples de codes binaires de longueur R. On dit que les couples (α, β) et (γ, δ) sont *non liés* (*découplés*) si le digit en r-ième position du code adopté $(1 \leqslant r \leqslant R)$ prend des valeurs opposées pour les couples (α, β) et (γ, δ). Dans le cas con-

traire on dit que les couples (α, β) et (γ, δ) sont *liés*. On démontre dans [42] le théorème suivant : *un automate dont les états sont codés par des codes binaires de longueur finie est exempt de courses si et seulement si pour deux transitions arbitraires* (a_m, a_s) *et* (a_h, a_l), $a_s \neq a_l$, *induites par un même signal d'entrée, les couples correspondants des codes d'états sont non liés.* On y trouve également un algorithme de codage des états des automates finis fondé sur ce théorème. L'idée maîtresse de cet algorithme est suffisamment simple : on examine successivement tous les couples de transitions pour lesquelles il

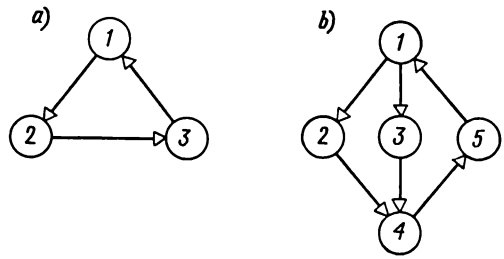


Fig. 5-13. Graphes n'admettant pas de codage par adjacence

existe au moins un signal d'entrée commun, et l'on donne aux digits correspondants des codes des valeurs telles que les couples des codes d'états soient non liés *).

Il existe enfin une méthode de codage particulière, dite *codage par adjacence*, ou *étude des adjacences*, qui permet d'éliminer les courses de façon certaine. Dans cette méthode deux états adjacents, c'est-à-dire liés par un arc sur le graphe de l'automate, sont codés par des collections qui ne diffèrent entre elles que par les états d'un seul élément de mémoire. On a proposé plusieurs algorithmes de codage par adjacence ; un algorithme est décrit dans [33].

Le codage par adjacence n'est pas toujours possible. On trouve dans [41] les conditions dans lesquelles ce codage peut être réalisé :

1. Le graphe de l'automate ne doit comporter aucun cycle au nombre impair de sommets.

2. Dans la totalité des chemins reliant entre eux deux états adjacents du second ordre on admet au plus deux états. Par états d'ordre i, en général, on entend deux états dont le chemin est constitué par i arêtes (d'orientation quelconque).

La figure 5-13 montre deux graphes qui ne satisfont pas ces conditions et qui ne peuvent pas être codés par adjacence.

En résumé, il existe quatre moyens d'éviter les courses :

*) Le lecteur trouvera une description détaillée de cet algorithme dans [5] bui constitue la première édition (en russe) du présent livre.

1) éléments de mémoire doublés;
2) choix judicieux de la durée de l'impulsion de synchronisation;
3) découplage des couples de transitions;
4) codage par adjacence.

5-6. Codage des états et complexité du circuit combinationnel

En étudiant la méthode canonique de synthèse structurelle des automates, on remarque que les différentes variantes de codage des états de l'automate conduisent à des expressions différentes des fonctions d'excitation de la mémoire et des fonctions des sorties; la complexité du circuit combinationnel de l'automate dépend donc notablement du procédé de codage retenu. A titre d'exemple, nous allons réaliser, avec des bascules du type D, l'automate abstrait S dont la table des transitions est résumée dans le tableau 5-4. Dans l'exemple considéré dans le paragraphe 5-3 (tableau 5-15), les états de l'automate ont été codés comme suit: $K(a_1) = 00$, $K(a_2) = 01$, $K(a_3) = 11$. Les fonctions d'excitation qui correspondent à un tel codage sont données par les expressions (5-5).

Tableau 5-31

Table des transitions de l'automate structuré

	01	10	00
00	10	—	01
01	00	01	—
10	10	00	00

Faisons le codage des états de S d'une autre manière: $K(a_1) = 01$, $K(a_2) = 10$, $K(a_3) = 00$. La table des transitions prend alors la forme du tableau 5-31.

Puisque, dans le cas d'utilisation des bascules D, la table de fonctions d'excitation est confondue avec la table des transitions nous pouvons déduire directement du tableau 5-31 des fonctions d'excitation beaucoup plus simples:

$$\varphi_1 = \bar{\tau}_1\tau_2\bar{x}_1\bar{x}_2 \vee \bar{\tau}_1\tau_2x_1\bar{x}_2 = 4 \vee 6,$$

$$\varphi_2 = \tau_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2 = 9 \vee 0.$$

Pour le codage des états dans l'exemple considéré, on a mis en œuvre l'algorithme suivant, qui permet de simplifier les fonctions d'excitation de l'automate construit à l'aide d'éléments de retard:

1. Associer à chaque état a_m de l'automate ($m = 1, \ldots, M$) un nombre entier N_m égal au nombre des transitions aboutissant à a_m (N_m est égal au nombre d'apparitions de a_m dans la table des transitions ou, si l'automate est défini graphiquement, au nombre des flèches aboutissant à a_m).

2. Ranger les nombres $N_1, \ldots, N_m, \ldots, N_M$ dans l'ordre des valeurs décroissantes.

3. Associer à l'état a_t ayant le plus grand N_t le code 00 . . . 0.

4. Associer aux R états suivants, ordonnés conformément au n° 2 ci-dessus (R étant le nombre des éléments de mémoire), les codes 00 . . . 01, 00 . . . 10, 00 . . . 100, . . ., 10 . . . 00 à une unité.

5. Pour le codage des R états suivants des $M - R - 1$ états qui restent, utiliser tous les codes contenant deux unités, puis trois unités, et ainsi de suite, jusqu'à ce que chaque état soit désigné par un code. Ce procédé de codage assure que les états auxquels il y a le plus de transitions seront codés par des combinaisons contenant le plus petit nombre d'unités possible.

Un procédé analogue peut être adopté pour le codage des signaux de sortie lors de la minimisation de la fonction des sorties. A titre d'exemple, nous allons ordonner les signaux de sortie de l'automate S d'après le nombre de leurs apparitions dans la table des sorties (tableau 5-5): w_3, trois fois; w_2, deux fois; w_1, une fois; w_4, une fois; u_1, deux fois; u_2, une fois.

Après le codage des signaux de sortie ordonnés (tableaux 5-32 et 5-33), la table des sorties de l'automate S prendra la forme du tableau 5-34.

Tableau 5-32

Codage des signaux de sortie du type 1

W	y_1y_2
w_3	00
w_2	01
w_1	10
w_4	11

Tableau 5-33

Codage des signaux de sortie du type 2

U	r
u_1	0
u_2	1

Tableau 5-34

Table des sorties après modification des codes des signaux de sortie

	0	1	0
	00	01	11
00	00	—	01
01	11	00	—
10	01	10	00

De ce tableau, nous obtenons directement les expressions des fonctions des sorties:

$$y_1 = \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \tau_1\tau_2x_1\bar{x}_2 = 1 \vee 6,$$

$$y_2 = \bar{\tau}_1\bar{\tau}_2\bar{x}_1x_2 \vee \bar{\tau}_1\bar{\tau}_2x_1\bar{x}_2 \vee \tau_1\tau_2\bar{x}_1\bar{x}_2 = 1 \vee 2 \vee 12,$$

$$r = \bar{\tau}_1\tau_2.$$

Remarquons que les expressions correspondantes (5-1, 5-3) obtenues en première variante de codage étaient beaucoup plus compliquées.

Beaucoup d'auteurs (voir [14, 26, 37, 63]), s'engageant dans la voie ouverte par J. Hartmanis et R. Stearns ([27, 58]), se sont attachés à inventer un codage tel que les fonctions d'excitation soient moins liées aux variables de boucle de retour $\tau_1, \ldots, \tau_R$. Hartmanis et Stearns ont montré que cette approche est étroitement liée à l'existence de partitions particulières de l'ensemble d'états de l'automate. En règle générale, un codage à dépendances réduites pour les fonctions d'excitation procure un schéma plus économique que les autres.

A titre d'exemple, considérons un automate dont la table des transitions est représentée par le tableau 5-35 *). Pour simplifier

Tableau 5-35

$\delta : A \times Z \rightarrow A$

	a_1	a_2	a_3	a_4	a_5	a_6
z_1	a_4	a_6	a_5	a_2	a_1	a_3
z_2	a_3	a_3	a_2	a_5	a_4	a_4

les choses, nous le construirons avec des bascules du type D, afin de pouvoir tirer les expressions des fonctions d'excitation directement de la table des transitions de l'automate structuré. Supposons que ses états internes et entrées soient codés conformément aux tableaux 5-36 et 5-37; le tableau 5-38 de l'automate structuré après

Tableau 5-36

Première variante de codage des états

	T_1	T_2	T_3
a_1	0	0	0
a_2	0	0	1
a_3	0	1	0
a_4	0	1	1
a_5	1	0	0
a_6	1	0	1

Tableau 5-37

Codage des signaux d'entrée

Z	X
z_1	0
z_2	1

*) L'exemple est tiré de [46].

Tableau 5-38

Table des transitions de l'automate structuré pour la première variante de codage

	000	001	010	011	100	101
0	011	101	100	001	000	010
1	010	010	001	100	011	011

simplification nous donne les expressions suivantes pour les fonctions d'excitation de la mémoire:

$$\varphi_1 = \bar{\tau}_1\bar{\tau}_2\tau_3\bar{x} \vee \tau_2\bar{\tau}_3\bar{x} \vee \tau_2\tau_3 x,$$

$$\varphi_2 = \bar{\tau}_2 x \vee \bar{\tau}_1\bar{\tau}_2\bar{\tau}_3 \vee \tau_1\tau_3,$$

$$\varphi_3 = \tau_1 x \vee \tau_2\bar{\tau}_3 x \vee \bar{\tau}_1\tau_3\bar{x} \vee \bar{\tau}_1\bar{\tau}_2\bar{x}.$$

Chaque variable φ_r ($r = 1, 2, 3$) est fonction de toutes les variables τ_1, τ_2, τ_3 et de la variable d'entrée x.

La deuxième variante de codage des états (tableaux 5-39 et 5-40) nous fournit des expressions plus simples:

$$\varphi_1 = \tau_1 x \vee \bar{\tau}_1\bar{x}, \quad \varphi_2 = \tau_3\bar{x}, \quad \bar{\varphi}_3 = \bar{\tau}_2\bar{\tau}_3.$$

On remarque que φ_1 est indépendante de τ_2 et de τ_3, tandis que φ_2 et φ_3 sont indépendantes de τ_1.

Tableau 5-39

Deuxième variante de codage des états

	T_1	T_2	T_3
a_1	1	1	0
a_2	1	0	1
a_3	1	0	0
a_4	0	0	0
a_5	0	0	1
a_6	0	1	0

On trouve dans la littérature, et notamment dans les monographies [28, 38, 63], des exposés suffisamment détaillés de l'approche mentionnée ainsi que des méthodes basées sur celle-ci. Il est clair que les méthodes de codage des états à dépendance fonctionnelle réduite sont étroitement liées au problème de décomposition d'un automate qui a été examiné dans le chapitre 4. D'autre part, la plupart des auteurs font remarquer que ces méthodes s'avèrent peu pratiques dans les cas où les états et transitions sont fort nombreux et où les automates sont partiels; ils indiquent également qu'il n'est pas facile de tenir compte en même temps de la complexité des fonctions d'excitation de la mémoire et de celle des fonctions des sorties. Pour ces raisons, au lieu d'examiner les méthodes citées, nous allons décrire un algorithme heuristique de codage des états proposé dans [48]. C'est un algorithme qui vise à minimiser le nombre

Tableau 5-40

Table des transitions de l'automate structuré pour la deuxième variante de codage

	110	101	100	000	001	010
0	000	010	001	101	110	100
1	100	100	101	001	000	000

total des changements d'état des éléments de mémoire pour l'ensemble des transitions de l'automate. Un tel critère contribue à diminuer la complexité des circuits qui réalisent les disjonctions aux entrées des éléments de mémoire, donc aussi à minimiser le circuit combinationnel de l'automate.

Soit $\Gamma(S)$ le graphe non orienté des transitions d'un automate S. Chaque sommet du graphe symbolise un état de l'automate. Deux sommets i et j sont liés par une arête chaque fois qu'il existe une transition de a_i à a_j ou de a_j à a_i. Désignons par $q(i, j)$ le nombre total des transitions de l'automate de l'état a_i à l'état a_j accompagnées de l'élaboration de signaux de sortie. A chaque arête (i, j) du graphe $\Gamma(S)$ associons un poids $p(i, j) = q(i, j) + q(j, i)$.

Introduisons une fonction

$$W = \sum_{(i,\ j) \in \Gamma(S)} p(i, j)\, d(i, j),$$

où $d(i, j)$ est le nombre de positions par lesquelles les codes $K(a_i)$ et $K(a_j)$ diffèrent (la *distance de Hamming* de $K(a_i)$ et $K(a_j)$). La fonction W ainsi définie peut servir de critère de complexité du circuit combinationnel de l'automate: la simplification du circuit combinationnel est d'autant plus poussée que W est moins grande.

L'algorithme comprend les pas suivants:

1. Construire la matrice

$$T = \begin{Vmatrix} \alpha_1 & \beta_1 \\ \cdot & \\ \alpha_R & \beta_R \end{Vmatrix},$$

qui comprend tous les couples d'indices (α_r, β_r), $r = 1, \ldots, R$, pour lesquels $p(\alpha_r, \beta_r) \neq 0$: on entend par là que l'automate peut passer de a_{α_r} à a_{β_r} ou de a_{β_r} à a_{α_r}, $\alpha_r < \beta_r$.

2. Ordonner les lignes de la matrice T. A cet effet, on construit une deuxième matrice M dont la première ligne sera constituée par le couple $(\alpha_{i1}, \beta_{i1})$ de plus grand poids $p(\alpha_{i1}, \beta_{i1})$. Parmi tous les couples admettant une composante en commun avec $(\alpha_{i1}, \beta_{i1})$,

on choisit un couple $(\alpha_{i2}, \beta_{i2})$ *) de plus grand poids et on l'inscrit dans la deuxième ligne de la matrice M. De tous les couples admettant une composante en commun avec au moins un des couples de la liste, on choisit le couple de plus grand poids et on l'inscrit dans M. Cette procédure est répétée autant de fois qu'il sera nécessaire pour que tous les couples de T soient inscrits dans M. Si deux couples présentent le même poids, on calcule les sommes des poids de leurs composantes (en désignant par le poids $p(\alpha_r)$ d'une composante α_r le nombre d'apparitions de α_r dans la matrice T) et l'on fait figurer dans la matrice M celui des couples dont la somme des poids des composantes est la plus élevée. Il est clair que, pour un automate connexe, un pareil rangement est toujours réalisable. On montre dans [48] qu'un rangement des couples suivant les poids conduit à une diminution de W en cas d'emploi de l'algorithme de codage qui sera exposé dans les alinéas suivants.

3. Coder les états de la première ligne de la matrice M comme suit :

$$K_{\alpha_{i1}} = 00 \ldots 00; \quad K_{\beta_{i1}} = 00 \ldots 01.$$

4. Rayer de M la première ligne qui correspond aux états codés $a_{\alpha_{i1}}$ et $a_{\beta_{i1}}$ **). On obtient une nouvelle matrice M'.

5. Le procédé de rangement adopté dans le n° 2 nous garantit que la première ligne de la matrice M' contient exactement un élément codé. Choisir l'élément non codé dans la première ligne de M' et le désigner par γ.

6. Construire une nouvelle matrice M_γ en relevant dans M' les lignes qui contiennent γ. Soit $B_\gamma = \{\gamma_1, \ldots, \gamma_f, \ldots, \gamma_F\}$ l'ensemble des éléments déjà codés de la matrice M_γ. Ils ont pour codes respectifs $K_{\gamma_1}, \ldots, K_{\gamma_f}, \ldots, K_{\gamma_F}$.

7. Pour chaque K_{γ_f} $(f = 1, \ldots, F)$ trouver l'ensemble $C^1_{\gamma_f}$ des codes adjacents à K_{γ_f} et qui ne sont pas encore affectés à un quelconque état de l'automate. Construire l'ensemble $D^1_\gamma = \bigcup_{f=1}^{F} C^1_{\gamma_f}$. Si $D^1_\gamma = \varnothing$, construire un nouvel ensemble $D^2_\gamma = \bigcup_{f=1}^{F} C^2_{\gamma_f}$, où $C^2_{\gamma_f}$ est l'ensemble des codes dont la distance de K_{γ_f} est égale à 2. Si $D^2_\gamma = \varnothing$ comme l'était D^1_γ, construire par analogie les ensembles $D^3_\gamma, \ldots, D^n_\gamma$, jusqu'à obtenir un $D^n_\gamma \neq \varnothing$ $(n = 1, 2, 3, \ldots)$. Soit $D^n_\gamma = \{K_{\delta_1}, \ldots, K_{\delta_g}, \ldots, K_{\delta_G}\}$.

8. Pour chaque K_{δ_g} calculer $d(\delta_g, \gamma_f)$, c'est-à-dire sa distance de tous les codes utilisés K_{γ_f} $(f = 1, \ldots, F)$.

---

*) Il est clair que $\{\alpha_{i1}, \beta_{i1}\} \cap \{\alpha_{i2}, \beta_{i2}\} \neq \varnothing$.

**) Dans le texte qui suit, au lieu de dire « élément α_r qui correspond à l'état codé a_{α_r} », nous dirons « élément codé α_r ».

9. Calculer $W_g = \sum_{\delta_g,\ \gamma_f \in \Gamma(S)} p(\delta_g, \gamma_f)\, d(\delta_g, \gamma_f) \quad g = 1, \ldots, G.$

10. Choisir dans D_γ^n un K_γ pour lequel $W_g = \min W_g$. Désigner cet élément γ (état a_γ) par le code K_γ.

11. Rayer de la matrice M' les lignes où les deux éléments sont codés : on obtient une nouvelle matrice M'. Si toutes les lignes de M' sont épuisées, passer au pas n° 12; s'il reste encore des lignes dans M', revenir au pas n° 5.

12. Calculer la fonction $W = \sum_{(i,j)\in\Gamma(S)} p(i, j)\, d(i, j)$. Passer au pas n° 13.

13. Fin.

Nous allons donner sans commentaire détaillé un exemple de codage des états d'un automate. Sa matrice T (qui correspond à la configuration de la figure 8-3) se présente comme suit :

$$T = \left\| \begin{array}{c|c} 12 & 3 \\ 13 & 1 \\ 14 & 1 \\ 15 & 2 \\ 23 & 2 \\ 24 & 1 \\ 34 & 1 \end{array} \right\|$$

On voit figurer à droite les poids $p(i, j)$.

Après le rangement des lignes de T conformément au pas n° 2, on obtient la matrice M.

$$K_1 = 000; \quad K_2 = 001.$$

Le codage sera illustré par la table de Karnaugh :

τ_1 \ $\tau_2\tau_3$	00	01	11	10
0	1	2		
1				

$$M = \left\| \begin{array}{c|c} 12 & 3 \\ 23 & 2 \\ 15 & 2 \\ 13 & 1 \\ 14 & 1 \\ 24 & 1 \\ 34 & 1 \end{array} \right\|; \quad M' = \left\| \begin{array}{c|c} 23 & 2 \\ 15 & 2 \\ 13 & 1 \\ 14 & 1 \\ 24 & 1 \\ 34 & 1 \end{array} \right\|; \quad \gamma = 3; \quad M_3 = \left\| \begin{array}{c|c} 23 & 2 \\ 13 & 1 \\ 34 & 1 \end{array} \right\|; \quad B_3 = \{1, 2\}.$$

$$C_1^1 = \{010,\ 100\};\quad C_2^1 = \{011,\ 101\};$$
$$D_3^1 = C_1^1 \cup C_2^1 = \{010,\ 100,\ 011,\ 101\}.$$
$$W_{010} = 1\cdot 1 + 2\cdot 2 = 5;\quad W_{100} = 1\cdot 1 + 2\cdot 2 = 5;$$
$$W_{011} = 1\cdot 2 + 2\cdot 1 = 4;\quad W_{101} = 1\cdot 2 + 2\cdot 1 = 4.$$

On choisit $K_3 = 011$.

τ_1 \ $\tau_2\tau_3$	00	01	11	10
0	1	2	3	
1				

$$M' = \left\| \begin{matrix} 15 \\ 14 \\ 24 \\ 34 \end{matrix} \right\| \begin{matrix} 2 \\ 1 \\ 1 \\ 1 \end{matrix};\quad \gamma = 5;\quad M_5 = \| 15 \|;\quad B_5 = \{1\}.$$

$$C_1^1 = \{100,\ 010\};\quad D_5^1 = C_3^1 = \{100,\ 010\}.$$

$W_{100} = 2\cdot 1 = 2$; $W_{010} = 2\cdot 1 = 2$. On choisit $K_5 = 100$.

τ_1 \ $\tau_2\tau_3$	00	01	11	10
0	1	2	3	
1	5			

$$M' = \left\| \begin{matrix} 14 \\ 24 \\ 34 \end{matrix} \right\| \begin{matrix} 1 \\ 1 \\ 1 \end{matrix};\ \gamma = 4;\ M_4 = \left\| \begin{matrix} 14 \\ 24 \\ 34 \end{matrix} \right\| \begin{matrix} 1 \\ 1 \\ 1 \end{matrix};\ B_4 = \{1,\ 2,\ 3\}.$$

$$C_1^1 = \{010\};\quad C_2^1 = \{101\};\quad C_3^1 = \{010,\ 111\};$$
$$D_4^1 = C_1^1 \cup C_2^1 \cup C_3^1 = \{010,\ 101,\ 111\}.$$
$$W_{010} = 1\cdot 1 + 1\cdot 2 + 1\cdot 1 = 4;\ W_{101} = 1\cdot 2 + 1\cdot 1 + 1\cdot 2 = 5;$$
$$W_{111} = 1\cdot 3 + 1\cdot 2 + 1\cdot 1 = 6.$$

On choisit $K_4 = 010$.

τ_1 \ $\tau_2\tau_3$	00	01	11	10
0	1	2	3	4
1	5			

La minimisation de la fonction W (dite fonction de coût) est également réalisable au moyen d'algorithmes de codage par itérations. Soit $K = (K_1, \ldots, K_m, \ldots, K_M)$ une solution du pro-

blème de codage des états, qui peut être obtenue soit par application de l'algorithme séquentiel décrit ci-dessus, soit par voie empirique. Au vecteur K correspond une valeur déterminée $W(K)$ de W. L'idée maîtresse de l'algorithme par itérations consiste à passer du vecteur K à un vecteur K' tel que $W(K') < W(K)$ (voir [48]). On y arrive par exemple en faisant toutes les permutations des coordonnées du vecteur K deux à deux, jusqu'à ce qu'on constate que W est devenue invariante par la permutation d'un couple quelconque K_i, K_j $(i \neq j)$.

On trouve dans [48] les résultats d'application expérimentale de quelques algorithmes de codage, qui témoignent de la haute efficacité de l'algorithme séquentiel.

CHAPITRE 6

SCHÉMAS OPÉRATORIELS D'ALGORITHMES

6-1. Microprogrammes de fonctionnement des réseaux séquentiels

Les systèmes d'automatismes et d'informatique sont représentés dans la plupart des cas de la façon classique, c'est-à-dire sous forme de composition de deux unités distinctes : l'unité de commande (A) et l'unité opérative (B) (fig. 6-1). Par exemple, si le système représenté est un ordinateur, son unité opérative comprend les ensembles de mémoire, les régistres, les additionneurs, les canaux de transmission d'information, les codeurs et les décodeurs, tandis que l'unité de commande assure la coordination des organes énumérés et l'enchaînement des opérations de traitement de l'information. La tâche principale de l'unité de commande consiste donc à élaborer une séquence temporelle de signaux de commande qui font exécuter une opération déterminée à l'unité opérative.

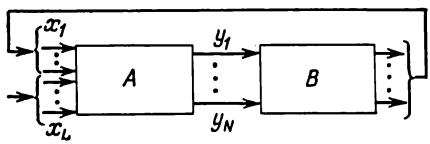


Fig. 6-1. Composition d'une unité de commande (A) et d'une unité opérative (B)

Dans certains cas, on adopte un mode de représentation hiérarchique du système séquentiel. Par exemple, en réalisant l'unité de commande centrale de l'ordinateur, on peut considérer l'unité arithmétique comme une partie de l'unité opérative ; à son tour, au sein de l'unité arithmétique, on prévoit alors un automate de commande et une partie opérative constituée par les registres, l'additionneur, le registre des codes d'opérations arithmétiques et le compteur de décalages.

Un acte élémentaire indivisible de traitement de l'information réalisé dans l'unité opérative pendant une période (ou temps) d'horloge sera appelé *micro-opération*. Soit $Y = \{y_1, \ldots, y_N\}$ l'ensemble des micro-opérations réalisées dans l'unité opérative. Elles sont commandées par des signaux $y_1, \ldots, y_N$ produits par l'unité de commande. Il y a intérêt à désigner le signal et la micro-opération par le même symbole : la micro-opération y_n ($n = 1, \ldots, N$) devient exécutable au moment où le signal 1 apparaît à la sortie y_n. Si l'unité B réalise plusieurs micro-opérations en même temps, l'ensemble de ces micro-opérations sera appelé *micro-instruction*. Par exemple, en

disant que $Y_t = \{y_{t1}, \ldots, y_{tU_t}\}$ est une micro-instruction, on sous-entend que toutes les micro-opérations $y_{t1}, \ldots, y_{tU_t}$ sont exécutées pendant une même période d'horloge ($y_{tu} \in Y$, $u = 1, \ldots, U_t$). Pour $U_t = 1$ la micro-instruction Y_t se réduit à une micro-opération unique. Le cas de $U_t = 0$ est également possible: il signifie que l'ensemble des micro-opérations constituant la micro-instruction est vide. Nous désignerons cette dernière par le symbole Y_0. L'exécution d'une telle micro-instruction est équivalente à l'absence de toute opération élémentaire exécutable. S'il s'agit d'un système séquentiel synchrone, on assimile la micro-instruction Y_0 à un temps vide, c'est-à-dire à un temps pendant lequel aucun signal ne parvient de l'unité de commande à l'unité opérative.

Soient $Y_1, \ldots, Y_T$ les microcommandes réalisables dans l'unité opérative. La succession des micro-instructions à réaliser est déterminée par les *fonctions de transition*: ce sont des fonctions booléennes α_{ij} ($i, j = 1, \ldots, T$) d'un ensemble de variables binaires $X = \{x_1, \ldots, x_L\}$ représentées par les variables d'entrée de l'unité de commande A. Il est naturel d'associer à chaque micro-instruction Y_i un ensemble de fonctions de transition $\{\alpha_{i1}, \ldots, \alpha_{iT}\}$, telles que si l'on a $\alpha_{it} = 1$ après l'exécution de la micro-instruction Y_i, la micro-instruction exécutable suivante est Y_t. Il est évident que l'ensemble des fonctions de transition à partir d'une même micro-instruction Y_i est *orthogonal* ($\alpha_{ij}\alpha_{it} = 0$ pour $j \neq t$) et *complet* ($\bigvee_{j=1}^{T} \alpha_{ij} = 1$). La première propriété veut dire qu'immédiatement après la micro-instruction donnée, vient une micro-instruction et une seule, exécutable pour des valeurs déterminées des variables d'entrée; la deuxième propriété signifie que cette dernière micro-instruction sera exécutée avec certitude.

L'ensemble des micro-instructions et des fonctions de transition forme ce qu'on appelle un *microprogramme*. Pour donner la description d'un microprogramme, on doit donc définir la succession des micro-instructions et des fonctions de transition, de façon à déterminer l'ordre d'exécution des micro-instructions. Pour la description des microprogrammes *), nous adopterons le langage des ordinogrammes, ou des schémas-graphes d'algorithmes.

6-2. Schémas-graphes d'algorithmes

Le *schéma-graphe d'algorithme* (SGA) est un graphe connexe orienté qui a un sommet *initial* (A_0), un sommet *final* (A_{fin}) et un en-

*) Dans les travaux de A. Tchébotarev, A. Michtchenko [59, 47], de F. Atstopas et coll. [1], la notion de microprogramme a un sens légèrement différent.

semble fini quelconque de sommets *conditionnels* ($P = \{p_1, \ldots, p_F\}$) et *opératoriels* ($A = \{A_1, \ldots, A_G\}$) (fig. 6-2).

Les sommets final, opératoriel ou conditionnel ont une entrée chacun; le sommet initial n'a aucune entrée. Le sommet initial ou

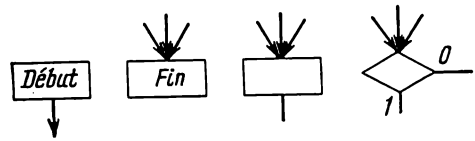


Fig. 6-2. Sommets d'un SGA

opératoriel a une sortie; le sommet conditionnel a deux sorties, marquées 1 et 0. Le sommet final n'a aucune sortie.

Tout schéma-graphe d'algorithme doit vérifier les conditions suivantes:

1. Les entrées et les sorties des sommets sont liées entre elles par des arcs qui sont toujours orientés de la sortie vers l'entrée.
2. Chaque sortie est liée exactement à une entrée.
3. Chaque entrée est liée à une sortie au moins.
4. Chaque sommet du graphe est intercalé au moins dans un chemin allant du sommet initial au sommet final.
5. Une des sorties d'un sommet conditionnel peut être liée à l'entrée de cette dernière, ce qui est inadmissible pour un sommet opératoriel. Un tel sommet conditionnel sera appelé parfois sommet à boucle de retour.
6. Dans chaque sommet conditionnel on inscrit un élément de l'ensemble $X = \{x_1, \ldots, x_l, \ldots, x_L\}$ appelé ensemble de *conditions logiques*. Les mêmes éléments de X peuvent être inscrits dans des sommets conditionnels différents.
7. Dans chaque sommet opératoriel on inscrit un opérateur (micro-instruction) Y_t, partie de l'ensemble $Y = \{y_1, \ldots, y_n, \ldots, y_N\}$ appelé ensemble de micro-opérations *):

$$Y_t = \{y_{t1}, \ldots, y_{tu}, \ldots, y_{tU_t}\}; \; y_{tu} \in Y, \; u = 1, \ldots, U_t.$$

Pour $U_t = 0$ on a $Y_t = \varnothing$; cela est admissible. De même, on peut inscrire les mêmes parties de Y dans des sommets opératoriels différents.

---

*) Quand on décrit un SGA, les termes *micro-instruction* et *opérateur* sont des synonymes. En décrivant des schémas-graphes d'algorithmes explicités (voir le paragraphe suivant), nous utiliserons seulement le terme micro-instruction, qui se rencontre plus souvent dans la littérature (voir par exemple [40]),.

S'il y a beaucoup d'arcs incidents vers l'intérieur à une même entrée du sommet, le graphisme risque de devenir trop embrouillé: nous représenterons alors les arcs entrant dans un sommet de la façon montrée sur la figure 6-3.

Un exemple de SGA à six sommets conditionnels et sept sommets opératoriels est représenté sur la figure 6-4.

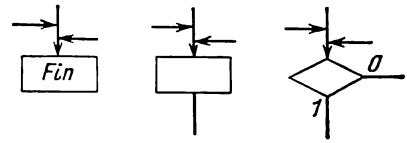


Fig. 6-3. Variantes de représentation des sommets de SGA

Supposons qu'on a inscrit dans chaque sommet opératoriel du SGA un des opérateurs $Y_1, \ldots, Y_T$. On peut alors identifier le sommet opératoriel à l'opérateur. Par exemple, on a inscrit l'opérateur Y_i dans le sommet A_i: pour désigner ce sommet, on peut alors utiliser le symbole Y_i au lieu de A_i.

Associons au sommet initial l'opérateur Y_0, et au sommet final, l'opérateur $Y_{fin} = Y_{T+1}$.

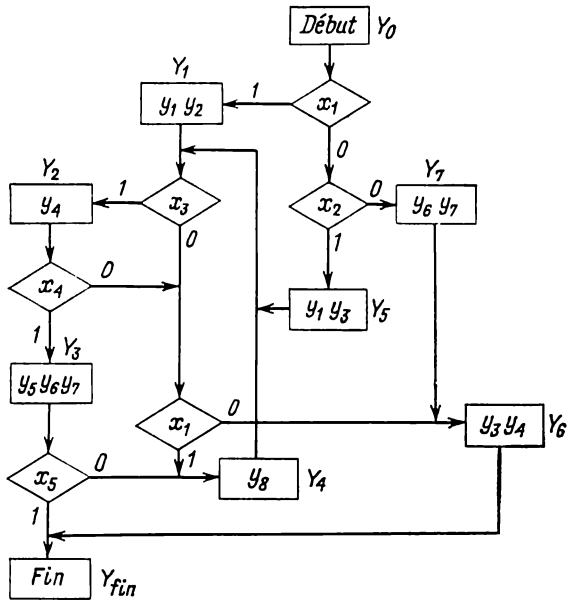


Fig. 6-4 Un exemple de SGA

Supposons que notre SGA comporte un chemin menant du sommet Y_i ($i = 0, 1, \ldots, T$) vers le sommet Y_j ($j = 1, \ldots, T, T + 1$):

$$Y_i p_{i1}^{e_{i1}} \ldots p_{ir}^{e_{ir}} \ldots p_{iR}^{e_{iR}} Y_j, \tag{6-1}$$

qui ne traverse que des sommets conditionnels $p_{i1}, \ldots, p_{iR}$; ici $e_{ir} \in \{0, 1\}$ est le symbole associé à la sortie du sommet conditionnel p_{ir} traversé par le chemin (6-1). Il est évident qu'à chaque chemin de ce type correspond une conjonction $\alpha_{ij} = x_{i1}^{e_{i1}} \ldots x_{iR}^{e_{iR}} = \bigwedge_{r=1}^{R} x_{ir}^{e_{ir}}$, où x_{ir} est la condition logique inscrite dans le sommet conditionnel p_{ir}; $x_{ir}^0 = \bar{x}_{ir}$, $x_{ir}^1 = x_{ir}$.

S'il existe entre les sommets Y_i et Y_j plusieurs (par exemple, H) chemins passant par des sommets conditionnels, alors α_{ij} est égal à la disjonction des conjonctions associées à tous les chemins, c'est-à-dire qu'on a alors $\alpha_{ij} = \bigvee_{h=1}^{H} \alpha_{ij}^h$, où α_{ij}^h est la conjonction associée au h-ième chemin allant de Y_i à Y_j. Nous appellerons α_{ij} *fonction de transition* de l'opérateur (micro-instruction) Y_i à l'opérateur (micro-instruction) Y_j.

Désignons par $\Delta_1, \ldots, \Delta_{2L}$ toutes les combinaisons possibles des valeurs des variables $x_1, \ldots, x_L$. Le processus d'*exécution* du SGA à partir de l'opérateur initial Y_0 sur une suite arbitraire infinie de collections $\Delta_{m1}, \ldots, \Delta_{mq}, \ldots$, sera décrit comme suit *).

On commence par inscrire l'opérateur Y_0.

1er pas. On donne aux variables $x_1, \ldots, x_L$ des valeurs qu'on prend dans la collection Δ_{m1}. De l'ensemble des fonctions de transition $\alpha_{01}, \ldots, \alpha_{0T}$ on tire la fonction α_{0i_1} ($i_1 \in \{1, \ldots, T\}$) qui prend la valeur 1 sur cette collection (α_{0i_1}, $(\Delta_{m1}) = 1$). On inscrit Y_{i_1} en ligne après Y_0:

$$Y_0 Y_{i_1}.$$

2e pas. On donne aux variables $x_1, \ldots, x_L$ des valeurs provenant de la collection Δ_{m2}. De l'ensemble des fonctions de transition $\alpha_{i_1 1}, \ldots, \alpha_{i_1 T}$ on tire une fonction $\alpha_{i_1 i_2}$ ($i_2 \in \{1, \ldots, T\}$) telle que $\alpha_{i_1 i_2}(\Delta_{m2}) = 1$. On inscrit Y_{i_2} en ligne après Y_{i_1}:

$$Y_0 Y_{i_1} Y_{i_2},$$

etc.

Supposons qu'on a obtenu, avant le q-ième pas, la ligne d'opérateurs

$$Y_0 Y_{i_1} Y_{i_2} \ldots Y_{i_{q-1}}.$$

Si sur Δ_{mq} une fonction $\alpha_{i_{q-1} i_q}$ prend la valeur 1 ($i_q \in \{1, \ldots, T\}$), on ajoute à la ligne l'opérateur Y_{i_q}. Si l'on a ensuite $\alpha_{i_{q-1}, T+1}(\Delta_{mq}) = 1$, on ajoute à la ligne, après $Y_{i_{q-1}}$, l'opérateur (final) Y_{T+1}, ce qui achève le processus d'exécution du SGA.

*) Il est supposé, dans le texte qui suit, que les valeurs des conditions logiques $x_1, \ldots, x_L$ ne peuvent changer qu'au moment d'exécution des opérateurs.

La ligne $Y_0Y_{i_1}Y_{i_2}\ldots Y_{i_{q-1}}Y_{i_q}$ s'appelle *valeur du* SGA *sur la séquence de collections* $\Delta_{m1}, \Delta_{m2}, \ldots, \Delta_{m,q-1}, \Delta_{mq}$ [66]. Il est clair que la procédure d'exécution du SGA sur la séquence donnée ne peut prendre fin que dans les deux cas suivants:

— toutes les collections ont été essayées (le dernier opérateur de la valeur du SGA est Y_{i_q});

— le sommet final est atteint (le dernier opérateur de la valeur du SGA est l'opérateur final Y_{T+1}).

C'est ainsi que le SGA montré sur la figure 6-1 prend sur la séquence de collections

x_1	x_2	x_3	x_4	x_5
1	0	1	0	1
1	1	0	1	1
1	0	1	1	0
0	1	1	0	1
1	0	1	0	0
0	1	0	1	1

la valeur égale à $Y_0Y_1Y_4Y_2Y_6Y_{\text{fin}}$.

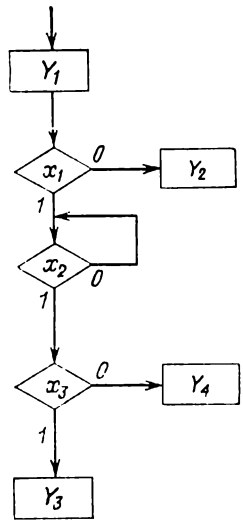


Fig. 6-5. Portion de SGA comportant un sommet conditionnel à boucle

Voyons comment se déroule l'exécution du SGA sur une séquence de collections arbitraire dans le cas où il y a des sommets conditionnels à boucle de retour (la portion correspondante du SGA est montrée sur la figure 6-5). Il y a entre les opérateurs Y_1 et Y_3 une infinité de chemins, ce qui fait que

$$\alpha_{13} = \alpha_{13}^1 \vee \alpha_{13}^2 \vee \ldots \vee \alpha_{13}^n \vee \ldots = x_1x_2x_3 \vee x_1\bar{x}_2x_2x_3 \vee$$
$$\vee\, x_1\bar{x}_2\bar{x}_2x_2x_3 \vee \ldots \vee x_1\bar{x}_2\cdot\ldots\cdot\bar{x}_2x_2x_3 \vee \ldots = x_1x_2x_3.$$

Puisque les conditions logiques correspondant à tous les chemins s'annulent à partir du deuxième chemin, α_{13} est pratiquement déterminé par le premier chemin. Nous ne prendrons donc en considération que les chemins qui, passant par un sommet à boucle, empruntent une sortie non rebouclée à l'entrée; nous admettrons qu'il existe, sur la figure 6-5, un seul chemin du type (6-1) allant du sommet Y_1 au sommet Y_3. Il ressort immédiatement de la figure 6-5 que

$$\alpha_{12} = \bar{x}_1, \quad \alpha_{13} = x_1x_2x_3, \quad \alpha_{14} = x_1x_2\bar{x}_3. \tag{6-2}$$

Supposons que l'opérateur Y_1 a été exécuté, pendant l'exécution du SGA, sur une séquence de collections et que la séquence en question se présente en ce moment comme suit :

$$1\ 0\ 1$$
$$1\ 1\ 1.$$

En regardant les expressions (6-2), on remarque qu'aucune fonction de transition à partir de l'opérateur Y_1 ne prend la valeur 1 sur la collection 101. On inscrit donc un opérateur vide Y_0 en ligne après Y_1 :

$$\ldots Y_1 Y_0.$$

Ceci fait, on met à l'essai la collection suivante (111) en cherchant la valeur que prennent sur cette dernière les fonctions de transition du même opérateur Y_1. On voit sans peine que $\alpha_{13} = 1$; on inscrit donc Y_3 après Y_0,

$$\ldots Y_1 Y_0 Y_3,$$

et ainsi de suite.

Si, par analogie au cas précédent, aucune des fonctions de transition n'avait pris la valeur 1 sur cette collection, nous aurions ajouté à Y_0 un nouvel opérateur vide Y_0, etc., jusqu'à ce qu'une fonction de transition prenne la valeur 1 sur une collection.

En effet, si les variables prennent des valeurs telles qu'aucune fonction de transition à partir du dernier opérateur exécuté n'est égale à 1, l'unité de commande n'envoie aucun signal d'exécution de micro-opération vers l'unité opérative (c'est la micro-instruction Y_0 qui est exécutée) ; le système est en attente.

Supposons qu'on a associé à chaque opérateur Y_t ($t = 0, 1, \ldots, T, T + 1$) du schéma-graphe un sous-ensemble B_t de conditions logiques dont les éléments peuvent changer pendant l'exécution de Y_t ; on a alors ce qu'on appelle une *distribution de décalages*. La distribution de décalages est une information complémentaire au SGA sur la logique de fonctionnement de l'unité ; elle peut s'avérer fort utile pour la simplification du SGA (cette question est examinée en détail dans le chapitre suivant).

Considérons l'exécution du SGA sur une séquence de collections de valeurs des conditions logiques

$$\Delta_{m1}, \ldots, \Delta_{mq}, \Delta_{m,q+1}.$$

Supposons que la collection Δ_{mq} soit directement antérieure à l'exécution de l'opérateur Y_{t_q} et la collection $\Delta_{m,q+1}$ lui est directement postérieure. Pour une distribution de décalages donnée, nous disons que la séquence de collections est *admissible* pour le SGA si chaque

collection $\Delta_{m,q+1}$ ou bien se confond avec Δ_{mq}, ou bien ne s'en distingue que par des valeurs des variables provenant de l'ensemble B_{i_q}.

Deux SGA, Γ_1 et Γ_2, sont équivalents (notation: $\Gamma_1 \equiv \Gamma_2$) pour une distribution de décalages donnée si leurs valeurs se confondent sur chaque séquence de collections admissible pour Γ_1 ou pour Γ_2 (voir [66]).

6-3. Schémas-graphes d'algorithmes explicités

L'étude d'un système commence généralement par l'établissement de son schéma-graphe d'algorithme explicité: c'est un graphe dont les sommets conditionnels et opératoriels contiennent au lieu des éléments d'ensembles X et Y les conditions logiques et les micro-opérations écrites sous forme explicite *).

Nous allons considérer à titre d'exemple l'algorithme d'exécution de l'opération de division dans une unité arithmétique (UA) parallèle qui travaille en virgule fixe. L'unité se compose d'un accumulateur (Σ) qui contient le dividende avant le commencement de l'opération, d'un registre Z contenant le diviseur et d'un registre Y qui reçoit le quotient après l'exécution de l'opération. Nous allons suivre l'exécution de l'opération de division d'après le schéma-graphe explicité de la figure 6-6.

On commence par déterminer le signe du quotient. Si le dividende et le diviseur ne sont pas de même signe (sign $Z \neq$ sign Σ), on inscrit 1 dans la position du signe du registre Y (sign Y): le quotient sera négatif. Si le signe est le même, cette phase est omise. Ceci fait, il se produit la remise à zéro du registre Y, du compteur de temps d'horloge (CT) et des positions du signe du registre Z et de l'accumulateur Σ. Ensuite, si le signe de l'accumulateur est égal à zéro (ce qui a toujours lieu au début de l'opération, à cause de la remise à zéro lors de la micro-instruction précédente), le diviseur est retranché du dividende: à cet effet, le code inverse du registre Z ($\Sigma := \Sigma + Z_{\text{inv}}$) est envoyé dans l'accumulateur. Si, au premier temps d'horloge (CT = 0), l'accumulateur contient après la soustraction un nombre positif (sign $\Sigma = 0$), le dividende est plus grand que le diviseur: il y a dépassement. Une unité est inscrite alors dans la bascule de dépassement (BD: = 1), et l'opération est terminée. Au contraire, si l'on a sign $\Sigma = 1$, il se produit l'incrémentation du compteur de temps d'horloge (CT: = CT + 1) et le décalage d'une position vers la gauche du contenu de l'accumulateur et du registre Y ($\Sigma := L_1(\Sigma)$, $Y := L_1(Y)$). Le deuxième temps d'hor-

*) V. Zakharov, D. Pospélov et V. Khazatski [67] décrivent d'une manière accessible quelques applications de la théorie des automates et des schémas opératoriels d'algorithmes à l'étude des systèmes de commande dans leur livre fort instructif.

loge commence. Si, à un certain temps d'horloge, l'accumulateur contient après la soustraction un nombre négatif, il se produit, au temps suivant, au lieu de la soustraction du contenu du registre Z

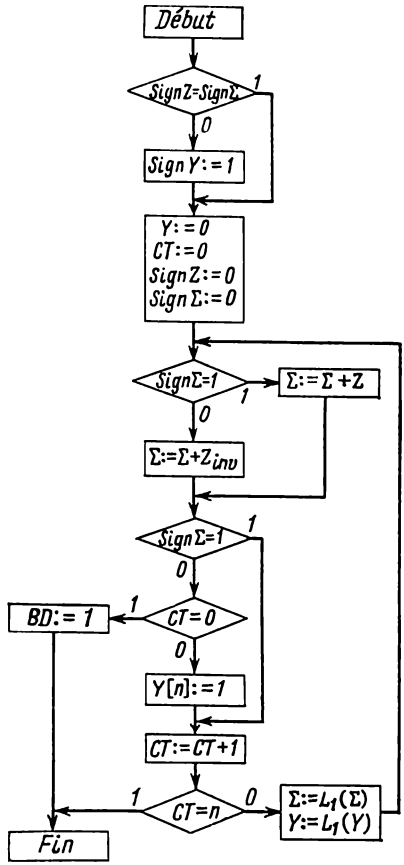


Fig. 6-6. SGA explicité de l'opération de division

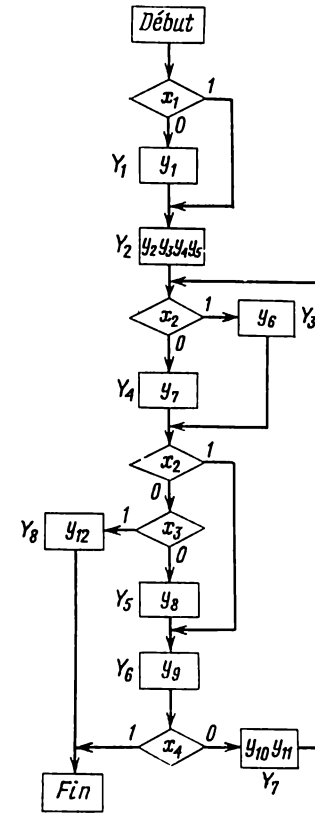


Fig. 6-7. SGA de l'opération de division

de celui de l'accumulateur, l'addition de ces deux quantités ($\Sigma := \Sigma + Z$), autrement dit, il se produit la division sans relèvement du reste. Si, à un certain temps d'horloge (à l'exception du premier), l'accumulateur contient un nombre positif (sign $\Sigma = 0$), une unité va s'inscrire en position n des poids faibles du résultat ($Y[n] := 1$). A l'expiration de n temps d'horloge ($CT = n$), l'opération prend fin. Le résultat de la division reste inscrit dans le registre Y.

Après l'établissement du SGA explicité, les conditions logiques et les micro-opérations sont codées par les symboles $x_1, \ldots, x_L$ et $y_1, \ldots, y_N$ respectivement :

Sommets conditionnels	*Sommets opératoriels*	
$x_1 : \operatorname{sign} \Sigma = \operatorname{sign} Z$,	$y_1 : \operatorname{sign} Y := 1$,	$y_7 : \Sigma := \Sigma + Z_{\mathrm{inv}}$,
$x_2 : \operatorname{sign} \Sigma = 1$,	$y_2 : Y := 0$,	$y_8 : Y[n] := 1$,
$x_3 : \mathrm{CT} = 0$,	$y_3 : \mathrm{CT} := 0$,	$y_9 : \mathrm{CT} := \mathrm{CT} + 1$,
$x_4 : \mathrm{CT} = n$.	$y_4 : \operatorname{sign} Z := 0$,	$y_{10} : \Sigma := L_1(\Sigma)$,
	$y_5 : \operatorname{sign} \Sigma := 0$,	$y_{11} : Y := L_1(Y)$,
	$y_6 : \Sigma := \Sigma + Z$,	$y_{12} : \mathrm{BD} := 1$.

Après un tel codage des sommets du SGA explicité de la figure 6-6, on obtient un SGA montré sur la figure 6-7.

6-4. Schémas logiques d'algorithmes

Les schémas-graphes d'algorithmes proposés par L. Kalouzhnine [32] sont une version graphique des schémas logiques d'algorithmes (SLA) introduits par A. Liapounov en 1953 pour la description des schémas-blocs des programmes d'ordinateurs [39]. A. Liapounov a proposé de mettre les algorithmes sous forme d'une ligne de symboles de longueur finie. Une telle ligne, qui se compose de symboles d'opérateurs et de conditions logiques et de flèches pointées vers le haut ou vers le bas marquées par des entiers naturels $\left(\overset{i}{\uparrow}, \overset{i}{\downarrow}, i = 1, 2, \ldots\right)$, s'appelle *schéma logique d'algorithme.*

Les SLA considérés dans le texte qui suit vérifient les conditions ci-après :

1. Ils contiennent un opérateur initial (Y_{in}) et un opérateur final (Y_{fin}).
2. Aucune flèche ne figure avant Y_{in} ni après Y_{fin}.
3. Chaque condition logique est suivie d'une flèche pointée vers le haut, ou montante.
4. Deux flèches pointées vers le bas, ou descendantes, n'ont jamais les mêmes marques (chiffres).
5. Pour toute flèche descendante $\overset{i}{\downarrow}$, il existe au moins une flèche montante.
6. Pour toute flèche montante, il existe toujours exactement une flèche descendante.

Les ensembles d'opérateurs et de conditions logiques seront désignés, par analogie aux schémas-graphes, par $Y = \{Y_1, \ldots, Y_T\}$ et $X = \{x_1, \ldots, x_L\}$ respectivement.

Nous allons décrire le fonctionnement d'un système séquentiel à l'aide de son SLA :

$$Y_{in}x_1 \overset{1}{\uparrow} Y_1 \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} Y_2Y_3 \overset{2}{\downarrow} Y_4Y_{fin}. \tag{6-3}$$

Ce schéma logique d'algorithme comprend les opérateurs initial et final (Y_{in} et Y_{fin}), quatre opérateurs Y_1 à Y_4 et deux conditions logiques. A l'opérateur initial correspond un état initial du système séquentiel, qui implique l'absence de toute micro-opération. Si le signal 1 vient s'appliquer à l'entrée x_1 du système qui se trouve à l'état initial ($x_1 = 1$), le système passe à un nouvel état qui correspond à l'exécution de l'opérateur *) Y_1 : le premier à droite après la condition logique x_1. Si $x_1 = 0$, on passe à la flèche montante $\overset{1}{\uparrow}$ et l'on cherche une flèche descendante marquée par le même chiffre $\left(\overset{1}{\downarrow}\right)$. Cette dernière est suivie de la condition logique x_2 et de la flèche $\overset{2}{\uparrow}$; ensuite on voit les opérateurs Y_2 et Y_3, et la flèche descendante $\overset{2}{\downarrow}$ est suivie de l'opérateur Y_4. Au niveau du comportement, une telle description se concrétise comme suit. Si $x_1 = 0$ et $x_2 = 1$ ($\bar{x}_1x_2 = 1$), le système passe de l'état initial à un nouvel état qui prévoit l'exécution de l'opérateur $Y_2\left(x_1 \overset{1}{\uparrow} \ldots \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} Y_2\right)$; si $x_1 = 0$ et $x_2 = 0$ ($\bar{x}_1\bar{x}_2 = 0$), c'est l'opérateur Y_4 qui est exécuté $\left(x_1 \overset{1}{\uparrow} \ldots \overset{1}{\downarrow} x_2 \overset{2}{\uparrow} \ldots \overset{2}{\downarrow} Y_4\right)$. L'opérateur Y_2 étant exécuté, quelles que soient les valeurs des conditions logiques, le système exécute l'opérateur Y_3 qui figure à droite de Y_2, puis l'opérateur Y_4, etc. Il est tout à fait évident que le SLA en question est équivalent au SGA de la figure 6-8.

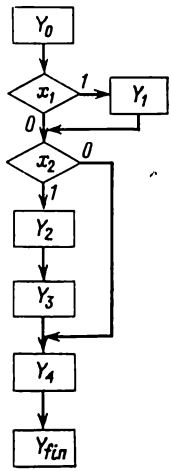


Fig. 6-8. SGA équivalent au SLA (6-3)

En écrivant des SLA, on a souvent intérêt à introduire des conditions logiques identiquement nulles, c'est-à-dire toujours fausses ; on les désigne par ω. Considérons par exemple une portion de SLA comprenant une condition logique ω :

$$Y_2\omega \overset{i}{\uparrow} \overset{j}{\downarrow} Y_3 \ldots \overset{i}{\downarrow} Y_t. \tag{6-4}$$

*) Dans le langage des SLA, comme dans celui des SGA, on entend par « opérateur » l'ensemble des micro-opérations exécutables en bloc.

Puisque $\omega = 0$, l'exécution de Y_2 conduit toujours directement à Y_t; quant à Y_3, il ne devient exécutable qu'en vertu de la flèche $\overset{j}{\downarrow}$.

On peut évidemment introduire pour les SLA la notion de chemin entre opérateurs, analogue à (6-1) pour les SGA, et celle de fonction de transition α_{ij} de l'opérateur Y_i à l'opérateur Y_j. L'on aura par exemple dans le SLA (6-3) $\alpha_{01} = x_1$; $\alpha_{02} = \bar{x}_1 x_2$; $\alpha_{04} = \bar{x}_1 \bar{x}_2$; $\alpha_{12} = x_2$; $\alpha_{14} = \bar{x}_2$; $\alpha_{23} = \alpha_{34} = \alpha_{4fin} = 1$; les autres fonctions de transition sont nulles. La valeur de SLA sur une séquence de collections admissible, la distribution des décalages, l'équivalence de deux SLA gardent la même signification que pour les schémas-graphes (voir § 6-2).

La théorie des SLA est exposée dans un ouvrage de fond devenu classique de You. Yanov [66]. Dans ce travail, l'auteur s'attache à étudier les questions d'équivalence de deux schémas logiques d'algorithmes et propose un système complet de transformations qui permet de trouver pour tout SLA un SLA équivalent.

Or, A. Erchov [16] remarquait que l'ouvrage de You. Yanov « n'a pas gagné un large auditoire, en premier lieu à cause du langage ésotérique, accessible plutôt aux spécialistes de logique mathématique qu'aux programmeurs ». Dans ce même travail, A. Erchov se donne pour but, pour les raisons évoquées, d'exprimer, dans un langage plus simple, les idées contenues dans [66]. Il constate que « si l'on utilise, au lieu des schémas logiques d'algorithmes écrits sous forme de lignes de symboles, des graphes orientés où les sommets symbolisent les opérateurs et les conditions logiques, et les flèches, les transferts de commande, cela conduit déjà à une simplification très sensible dans la plupart des cas; on arrive en particulier à simplifier le système de transformations, ne laissant que six règles au lieu de dix-sept » *).

C'est V. Lasarev qui, avec ses collègues, a proposé d'utiliser les schémas logiques d'algorithmes comme langage de description des automates microprogrammés; les résultats de leurs études dans ce domaine sont résumés dans la monographie [38] **).

Examinons l'algorithme de passage du SGA au SLA en prenant le schéma-graphe de la figure 6-4.

1. Marquer par 1, 2, . . ., s les entrées de chaque sommet du SGA auquel aboutissent plus d'une flèche, ainsi que l'entrée du som-

---

*) Ce n'est donc pas les prédilections de l'auteur du présent ouvrage mais aussi les considérations évoquées plus haut qui l'ont incité à adopter dans son livre le langage des schémas-graphes pour la destription des algorithmes de fonctionnement des systèmes séquentiels.

**) Les questions de transformation des schémas opératoriels d'algorithmes et de synthèse des automates sont aussi traitées dans une série de travaux de V. Barachenkov et coll. (voir par exemple [3, 57]).

met final même s'il n'y aboutit qu'une seule flèche (fig. 6-9). Pour le SGA de la figure 6-9 on a $s = 5$.

2. Inscrire l'opérateur Y_{in} qui correspond au sommet initial du SGA.

3. Si l'entrée du sommet qui suit le sommet initial a été marquée précédemment par un symbole j, faire suivre l'opérateur Y_{in} d'une flèche descendante $\overset{j}{\downarrow}$. Ce cas ne se présente pas dans l'exemple considéré.

4. Si le sommet initial du SGA est suivi d'un sommet opératoriel marqué par l'opérateur Y_t, ajouter Y_t à droite de Y_{in}. Si le sommet

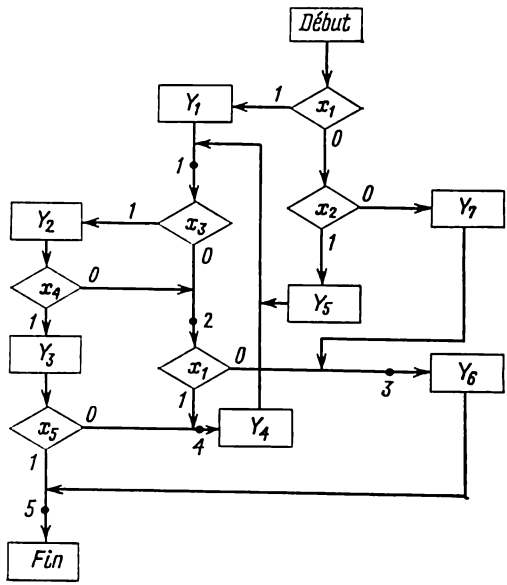


Fig. 6-9. Marquage du SGA en vue de la construction d'un SLA équivalent

initial est suivi d'un sommet conditionnel marqué par une condition logique x_m, ajouter x_m à droite de Y_{in}. A droite de x_m mettre une flèche montante surmontée d'un i si telle est la marque de l'entrée du sommet suivant auquel est reliée la sortie « 0 » du sommet conditionnel considéré. Si la marque est inexistante, mettre à droite de x_m une flèche montante muette. Dans notre exemple

$$Y_{in}x_1 \uparrow .$$

5. Si le sommet considéré dans le n° 4 est opératoriel et si l'entrée du sommet suivant a été marquée par un p dans le n° 1, inscrire

après Y_t une flèche descendante $\overset{p}{\downarrow}$, puis inscrire le contenu du sommet qui suit le sommet opératoriel d'opérateur Y_t. Si le sommet considéré dans le n° 4 est conditionnel de condition logique x_m, inscrire à la suite de sa flèche montante une flèche descendante $\overset{p}{\downarrow}$, au cas où l'entrée du sommet suivant, à laquelle est liée la sortie « 1 » du sommet conditionnel, est marquée par p. Dans le cas contraire, aucune flèche ne sera introduite. Ceci fait, écrire à droite de x_m le contenu du sommet qui suit la sortie « 1 » du sommet conditionnel considéré. Dans notre exemple

$$Y_{1n}\ x_1 \uparrow Y_1.$$

6. Continuer de la sorte jusqu'à ce que:

a) une flèche descendante $\overset{s}{\downarrow}$, où s est la marque du sommet final, vienne se placer après un opérateur quelconque Y_t: on inscrit alors après Y_t une condition logique identiquement fausse ω suivie d'une flèche montante $\overset{s}{\uparrow}$:

$$\ldots Y_t \omega \overset{s}{\uparrow};$$

b) une flèche descendante $\overset{q}{\downarrow}$ déjà figurant dans la ligne se présente après un opérateur quelconque Y_t: au lieu d'inscrire $\overset{q}{\downarrow}$ après Y_t, on met alors une condition logique identiquement fausse ω suivie d'une flèche montante $\overset{q}{\uparrow}$ $\left(\omega \overset{q}{\uparrow}\right)$;

c) une flèche descendante $\overset{s}{\downarrow}$, où s est la marque du sommet final, vienne après une condition logique x_m suivie de sa flèche montante $(x_m\uparrow)$: on inscrit alors après x_m une condition logique identiquement fausse ω suivie de la flèche montante $\overset{s}{\uparrow}$:

$$x_m \uparrow \omega \overset{s}{\uparrow};$$

d) après une condition logique x_m suivie de sa flèche montante $(x_m\uparrow)$, on voie apparaître une flèche descendante $\overset{q}{\downarrow}$ (où q est la marque de l'entrée du sommet auquel est reliée la sortie « 1 » du sommet conventionnel qui correspond à cette condition x_m), alors qu'il y a déjà une flèche descendante $\overset{q}{\downarrow}$ à gauche de x_m. Dans ce cas, au

lieu de la flèche descendante $\overset{q}{\downarrow}$ après x_m, on inscrit une condition logique identiquement fausse ω suivie d'une flèche montante $\overset{q}{\uparrow}$:

$$\ldots x_m \uparrow \omega \overset{q}{\uparrow}.$$

Dans notre exemple, la première ligne se termine avec l'apparition de la flèche descendante $\overset{5}{\downarrow}$ à la suite de $x_5 \overset{4}{\uparrow}$. Ici 5 est la marque du sommet final (voir condition 6, c):

$$Y_{1n}x_1 \uparrow Y_1 \overset{1}{\downarrow} x_3 \overset{2}{\uparrow} Y_2x_4 \overset{2}{\uparrow} Y_3x_5 \overset{4}{\uparrow} \overset{5}{\downarrow}.$$

En écrivant ω et une flèche montante $\overset{5}{\uparrow}$ après $x_5 \overset{4}{\uparrow}$, on obtient:

$$Y_{1n}x_1 \uparrow Y_1 \overset{1}{\downarrow} x_3 \overset{2}{\uparrow} Y_2x_4 \overset{2}{\uparrow} Y_3x_5 \overset{4}{\uparrow} \omega \overset{5}{\uparrow}. \tag{6-4}$$

7. Trouver dans (6-4) la première flèche montante muette et inscrire au-dessus de cette flèche le nombre $s + 1$:

$$Y_{1n}x_1 \overset{6}{\uparrow} Y_1 \ldots$$

8. Commencer une nouvelle ligne en écrivant une flèche descendante $\overset{s+1}{\downarrow}$. Ensuite, inscrire le contenu du sommet auquel est liée la sortie « 0 » du sommet conditionnel de condition logique x_1 (au-dessus de la flèche postérieure à x_1 nous avons inscrit $s + 1 = 6$):

$$\overset{6}{\downarrow} x_2 \uparrow.$$

9. Procéder comme dans la première ligne, à ceci près que toutes les conditions du n° 6 seront vérifiées pour toutes les lignes déjà écrites. On obtient finalement les trois lignes suivantes:

$$Y_{1n}x_1 \overset{6}{\uparrow} Y_1 \overset{1}{\downarrow} x_3 \overset{2}{\uparrow} Y_2x_4 \overset{2}{\uparrow} Y_3x_5 \overset{4}{\uparrow} \omega \overset{5}{\uparrow}, \tag{6-5}$$

$$\overset{6}{\downarrow} x_2 \overset{7}{\uparrow} Y_5\omega \overset{1}{\uparrow}, \tag{6-6}$$

$$\overset{7}{\downarrow} Y_7 \overset{3}{\downarrow} Y_6\omega \overset{5}{\uparrow}. \tag{6-7}$$

La première, la deuxième et la troisième lignes prennent fin quand les conditions 6,c, 6,b et 6,a ont été vérifiées respectivement.

10. Après le n° 9, il ne reste aucune flèche montante muette. Chercher alors un r ($1 \leqslant r \leqslant s$, où s est le nombre des marques faites

sur le SGA dans le n° 1) tel qu'il n'existe pas de flèche descendante $\overset{r}{\downarrow}$ dans les lignes écrites. Ceci fait, commencer une nouvelle ligne en écrivant d'abord une flèche $\overset{r}{\downarrow}$, puis le contenu du sommet marqué par r. Dans notre exemple $r = 2$:

$$\overset{2}{\downarrow} x_1 \overset{3}{\uparrow} \overset{4}{\downarrow} Y_4 \omega \overset{1}{\uparrow} . \tag{6-8}$$

11. Si possible, procéder comme décrit ci-dessus.

12. La dernière ligne pour n'importe quel schéma-graphe se réduit à une flèche descendante $\overset{s}{\downarrow}$, où s est la marque du sommet final, et au symbole de l'opérateur final. Dans notre exemple

$$\overset{5}{\downarrow} Y_{\text{fin}}. \tag{6-9}$$

13. A partir des lignes (6-5) à (6-9), former une ligne unique en mettant la première (6-5) au commencement et la dernière (6-9) à la fin, et en disposant les autres lignes dans un ordre arbitraire:

$$Y_{\text{in}} x_1 \overset{6}{\uparrow} Y_1 \overset{1}{\downarrow} x_3 \overset{2}{\uparrow} Y_2 x_4 \overset{2}{\uparrow} Y_3 x_5 \overset{4}{\uparrow} \omega \overset{5}{\uparrow} \overset{6}{\downarrow} x_2 \overset{7}{\uparrow} Y_5 \omega \overset{1}{\uparrow} \overset{7}{\downarrow} Y_7 \overset{3}{\downarrow} Y_6$$
$$\omega \overset{5}{\uparrow} \overset{2}{\downarrow} x_1 \overset{3}{\uparrow} \overset{4}{\downarrow} Y_4 \omega \overset{1}{\uparrow} \overset{5}{\downarrow} Y_{\text{fin}}. \tag{6-10}$$

Le procédé de construction même de SLA permet de voir que le SLA en question est équivalent au SGA de la figure 6-9. On montre tout aussi facilement qu'un SLA ainsi construit comporte autant d'opérateurs et de conditions logiques (sans compter les conditions identiquement fausses) qu'il y a de sommets opératoriels et conditionnels dans le schéma-graphe initial.

Quant à l'algorithme de passage du SLA au SGA, il est immédiat. Le SGA correspondant admet un sommet initial et un sommet final; le nombre de ses sommets opératoriels est égal au nombre des opérateurs du SLA, et le nombre de ses sommets conditionnels, à celui des conditions logiques du SLA (sans compter les conditions identiquement fausses). La sortie du sommet initial du SGA sera reliée par un arc à l'entrée du sommet qui correspond, dans le SLA, à l'opérateur ou à la condition logique se trouvant le plus près à droite de Y_{in}. Les sommets opératoriels du schéma-graphe sont munis d'arcs d'une façon analogue. La sortie « 1 » d'un sommet du SGA qui correspond à la condition logique x_m du SLA est reliée à l'entrée du sommet qui correspond à l'opérateur ou à la condition logique se trouvant immédiatement à droite de x_m. Si une condition logique x_m dans SLA est suivie d'une flèche montante $\overset{i}{\uparrow}$, la sortie « 0 » du sommet conditionnel correspondant à x_m sera reliée à l'en-

trée du sommet qui correspond à l'opérateur ou à la condition logique précédé(e) par une flèche descendante $\overset{i}{\downarrow}$. Si un opérateur Y_t du SLA est suivi d'une condition logique identiquement fausse ω avec une flèche montante $\overset{i}{\uparrow}$ $\left(\omega \overset{i}{\uparrow}\right)$, la sortie du sommet opératoriel correspondant à Y_t est à relier à l'entrée du sommet qui correspond à l'opérateur ou à la condition logique figurant après la flèche descendante $\overset{i}{\downarrow}$. De même, si une condition logique x_m avec flèche montante ($x_m \uparrow$) est suivie de $\omega \overset{i}{\uparrow}$, on doit relier la sortie «1» du sommet conditionnel correspondant à x_m à l'entrée du sommet qui correspond à l'opérateur ou à la condition logique figurant après la flèche descendante $\overset{i}{\downarrow}$.

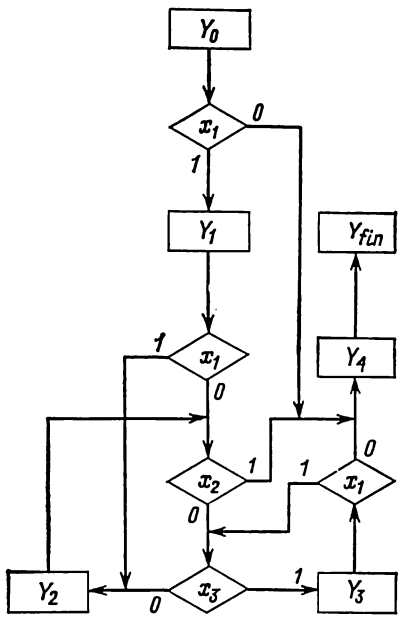


Fig. 6-10. Un exemple de SGA

Si l'on se propose de passer du SLA (6-10) à un SGA, on obtient le schéma-graphe représenté sur la figure 6-9.

6-5. Formules de transition

Les algorithmes admettent une autre représentation opératorielle, effectuée à l'aide des formules de transition. Supposons par exemple que l'opérateur Y_1 vient d'être exécuté ; l'opérateur exécutable suivant sera alors Y_2, Y_4 ou Y_6 (voir fig. 6-4), à condition que la fonction de transition correspondante prenne la valeur 1. Cet état de choses peut être résumé par la formule

$$Y_1 \rightarrow x_3 Y_2 \vee \bar{x}_3 x_1 Y_4 \vee \bar{x}_3 \bar{x}_1 Y_6$$

dans laquelle $\alpha_{12} = x_3$, $\alpha_{14} = \bar{x}_3 x_1$, $\alpha_{16} = \bar{x}_3 \bar{x}_1$, tandis que toutes les autres fonctions de transition à partir de l'opérateur Y_1 sont nulles.

Dans le cas général la *formule de transition* d'un opérateur Y_i se présente sous la forme

$$Y_i \rightarrow \bigvee_{t=1}^{T+1} \alpha_{it} Y_t. \qquad (6\text{-}11)$$

Ici α_{it} est la fonction de transition de l'opérateur Y_i à l'opérateur Y_t ($t = 1, \ldots, T + 1$); c'est une fonction booléenne des variables binaires $x_1, \ldots, x_L$. La quantité $\alpha_{it}Y_t$ porte le nom de *fonction booléenne marquée*; elle peut prendre les valeurs

$$\alpha_{it}Y_t = \begin{cases} Y_t & \text{si} \quad \alpha_{it} = 1, \\ 0 & \text{si} \quad \alpha_{it} = 0. \end{cases}$$

Les fonctions booléennes figurant dans les fonctions marquées se prêtent à toutes les transformations de l'algèbre de la logique. On trouve dans [49] des règles qui, ajoutées au système complet d'axiomes de l'algèbre de la logique, forment la liste des règles de transformation identique des fonctions marquées:

1. $(\alpha \vee \beta)\, Y_i = \alpha Y_i \vee \beta Y_i$.
2. $\alpha\beta Y_i \vee \alpha\gamma Y_j = \alpha\, (\beta Y_i \vee \gamma Y_j)$.
3. Si $\alpha \equiv \beta$ (α est équivalent à β), on a $\alpha Y_i = \beta Y_i$.

Dans ces expressions α, β et γ sont des fonctions booléennes des variables $x_1, \ldots, x_L$.

L'ensemble des formules de transition pour tous les $i = 0, 1, \ldots, T$ forme le *système de formules de transition*. Pour le SGA montré sur la figure 6-10, ce système se présente comme suit:

$$\left.\begin{aligned} &Y_0 \to x_1 Y_1 \vee \bar{x}_1 Y_4, \\ &Y_1 \to x_1 Y_2 \vee \bar{x}_1\bar{x}_2\bar{x}_3 Y_2 \vee \bar{x}_1\bar{x}_2 x_3 Y_3 \vee \bar{x}_1 x_2 Y_4, \\ &Y_2 \to \bar{x}_2\bar{x}_3 Y_2 \vee \bar{x}_2 x_3 Y_3 \vee x_2 Y_4, \\ &Y_3 \to x_1\bar{x}_3 Y_2 \vee x_1 x_3 Y_3 \vee \bar{x}_1 Y_4, \\ &Y_4 \to Y_{\text{fin}}. \end{aligned}\right\} \qquad (6\text{-}12)$$

La représentation d'une formule de transition sous la forme

$$Y_i \to x_l A \vee \bar{x}_l B, \qquad (6\text{-}13)$$

où $x_l \in \{x_1, \ldots, x_L\}$ et A, B sont deux formules de transition partielles indépendantes de x_l, s'appelle *réduction* (*développement*) de la formule de transition suivant la variable x_l. Ecrivons, par exemple, la formule de transition pour l'opérateur Y_1 de (6-12) développée suivant la variable x_1:

$$Y_1 \to x_1 Y_2 \vee \bar{x}_1 (\bar{x}_2\bar{x}_3 Y_2 \vee \bar{x}_2 x_3 Y_3 \vee x_2 Y_4). \qquad (6\text{-}14)$$

Si la formule de transition comporte des termes indépendants de x_l, on doit, avant de faire la réduction suivant x_l, multiplier ces termes par la quantité $(x_l \vee \bar{x}_l)$: de cette façon, n'importe quelle formule de transition se laisse développer suivant n'importe laquelle

de ses variables. Par exemple, le développement de la formule de transition de Y_1 suivant x_2 s'écrit

$$Y_1 \to x_1 (x_2 \vee \bar{x}_2) Y_2 \vee \bar{x}_1\bar{x}_2\bar{x}_3 Y_2 \vee \bar{x}_1\bar{x}_2 x_3 Y_3 \vee \bar{x}_1 x_2 Y_4 =$$
$$= x_2 (x_1 Y_2 \vee \bar{x}_1 Y_4) \vee \bar{x}_2 (x_1 Y_2 \vee \bar{x}_1\bar{x}_3 Y_2 \vee \bar{x}_1 x_3 Y_3). \quad (6\text{-}15)$$

Dans la relation (6-13), on peut continuer à développer les expressions A et B suivant les autres variables; ce développement prendra cependant fin au moment où l'on aura dans les parenthèses intérieures des expressions du type

$$(x_p Y_m \vee \bar{x}_p Y_n), \quad (6\text{-}16)$$

dans lesquelles

$$x_p \in \{x_1, \ldots, x_L\}, \; Y_m, \; Y_n \in \{Y_1, \ldots, Y_{T+1}\}.$$

Une formule de transition ainsi obtenue est dite *parenthétisée*, et le système de formules, un *système parenthétisé*. Les expressions du type (6-16) sont appelées élémentaires.

Faisons la parenthétisation du système (6-12):

$$\left.\begin{array}{l}
Y_0 \to x_1 Y_1 \vee \bar{x}_1 Y_4, \\
Y_1 \to x_1 Y_2 \vee \bar{x}_1 (\bar{x}_2 (\bar{x}_3 Y_2 \vee x_3 Y_3) \vee x_2 Y_4), \\
\qquad\qquad \cdots\cdots\cdots \\
Y_2 \to \bar{x}_2 (\bar{x}_3 Y_2 \vee x_3 Y_3) \vee x_2 Y_4, \\
\qquad\qquad \cdots\cdots\cdots \\
Y_3 \to x_1 (\bar{x}_3 Y_2 \vee x_3 Y_3) \vee \bar{x}_1 Y_4, \\
\qquad\qquad \cdots\cdots\cdots \\
Y_4 \to Y_{\text{fin}}.
\end{array}\right\} \quad (6\text{-}17)$$

La parenthétisation de la formule de transition (6-15) nous donne

$$Y_1 \to x_2 (x_1 Y_2 \vee \bar{x}_1 Y_4) \vee \bar{x}_2 (x_1 Y_2 \vee \bar{x}_1 (\bar{x}_3 Y_2 \vee x_3 Y_3)). \quad (6\text{-}18)$$

Il est évident qu'à chaque expression (6-16) correspond, sur le SGA, un sous-graphe montré sur la figure 6-11. Une fois parenthétisées, les formules de transition permettent de passer sans difficulté au SGA : il suffit de construire des chaînes de sommets conditionnels conformément à la séquence de développement de la formule de transition. On voit sur la figure 6-12 le sous-graphe du SGA qui correspond aux formules de transition parenthétisées (6-17) pour les opérateurs Y_1 et Y_2. Il est évident que deux expressions identiques donnent lieu à un même sous-graphe du SGA (ces expressions sont mises en relief par des points dans (6-17)).

Proposons-nous maintenant de construire le sous-graphe du SGA pour Y_1 et Y_2 en prenant, au lieu de la formule de transition paren

thétisée (6-17) pour Y_1, l'expression (6-18) : le sous-graphe qui en résulte (fig. 6-13) aura deux sommets conditionnels de plus. Il est

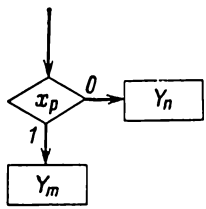


Fig. 6-11. Sous-graphe de SGA correspondant à l'expression (6-16)

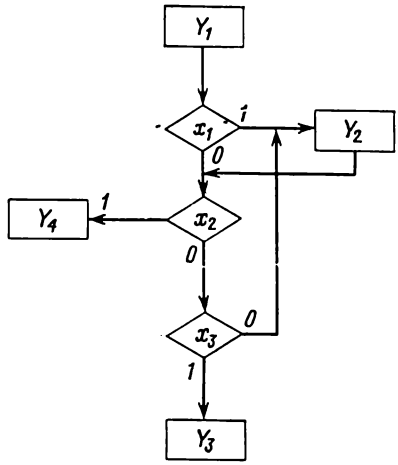
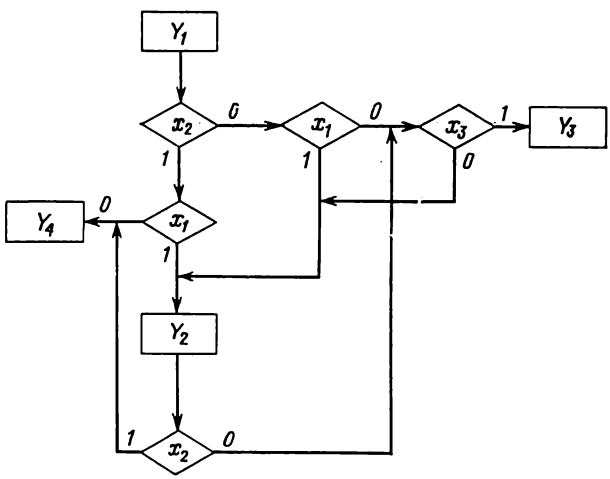


Fig. 6-12. Sous-graphe de SGA pour Y_1 et Y_2 construit d'après les expressions (6-17)

clair que le nombre des sommets conditionnels du SGA dépend de l'ordre de développement des formules de transition. En outre, le



Fir. 6-13. Sous-graphe de SGA pour Y_1 et Y_2 construit en tenant compte de (6-18)

SGA sera d'autant plus compliqué que le système de formules de transition contient plus de parenthèses renfermant des expressions

identiques. On conçoit qu'une « bonne » réduction n'est possible que sur l'ensemble de formules de transition. La question de minimisation des sommets conditionnels sur le schéma-graphe sera discutée en détail dans le septième chapitre.

6-6. Schémas matriciels d'algorithmes

Soit un SGA Γ muni d'un sommet initial Y_0, d'un sommet final Y_{fin} et de T sommets opératoriels auxquels sont associés des opérateurs différents $Y_1, \ldots, Y_T$. Le *schéma matriciel d'algorithme* (SMA) M correspondant au SGA Γ est une matrice carrée dont les lignes sont marquées par les symboles $Y_0, Y_1, \ldots, Y_T$, et les colonnes, par les symboles $Y_1, Y_2, \ldots, V_T, Y_{\text{fin}}$. A l'intersection d'une ligne Y_i et d'une colonne Y_j, on y trouve la fonction de transition α_{ij} de l'opérateur Y_i à l'opérateur Y_j. Le SMA correspondant au SGA de la figure 6-10 est donné dans le tableau 6-1. Pour simplifier la représentation, on omet d'inscrire dans la matrice les fonctions de transition identiquement nulles.

Tableau 6-1

SMA correspondant au SGA de la figure 6-10

	Y_1	Y_2	Y_3	Y_4	Y_{fin}
Y_0	x_1			$\bar{x}_1$	
Y_1		x_1 $\bar{x}_1\bar{x}_2\bar{x}_3$	$\bar{x}_1\bar{x}_2x_3$	$\bar{x}_1x_2$	
Y_2		$\bar{x}_2\bar{x}_3$	$\bar{x}_2x_3$	x_2	
Y_3		$x_1\bar{x}_3$	x_1x_3	$\bar{x}_1$	
Y_4					1

La procédure de passage du SGA au SMA est immédiate: on cherche sur le SGA toutes les fonctions de transition et on les range dans les cases correspondantes de la matrice. Le passage du SMA au SGA consiste à écrire successivement toutes les formules de tran-

sition, à parenthétiser ce système et à tracer le SGA correspondant *).

Comme dans le cas des schémas-graphes, on peut définir la *valeur que prennent le* SMA *et le système de formules de transition* sur une séquence quelconque de collections de valeurs des variables $x_1, \ldots, x_L$.

*) M. Gavrilov et ses collègues ont proposé dans [19, 20] un langage des schémas opératoriels d'algorithmes parallèles à mémoire pour la description du comportement des systèmes séquentiels.

CHAPITRE 7

TRANSFORMATION DES SCHÉMAS OPÉRATORIELS D'ALGORITHMES

7-1. Minimisation des sommets conditionnels sur le schéma-graphe d'algorithme

Nous lisons dans [15] que le système complet de transformations identiques des schémas logiques d'algorithmes (SLA) proposé par You. Yanov dans [66] permet d'obtenir un SLA avec un minimum de conditions logiques, mais le seul moyen d'y arriver consiste dans l'examen complet de tous les SLA possibles, faute d'algorithme de minimisation du SLA par transformations successives. Dans le présent paragraphe sera décrit un algorithme de minimisation du nombre de conditions logiques proposé par l'auteur dans [4]; bien que cet algorithme soit adapté en l'occurrence aux schémas-graphes, tous les raisonnements restent applicables aux schémas logiques d'algorithmes.

Le problème se pose dans le cas général comme suit. Soit un SGA $\Gamma\,(Y_0, Y_{\text{fin}}, Y, P)$, où Y_0 est le sommet initial, Y_{fin} le sommet final, $Y = \{Y_1, \ldots, Y_T\}$ l'ensemble des sommets opératoriels distincts (chaque sommet étant identifié à son propre opérateur), et $P = \{p_1, \ldots, p_R\}$ l'ensemble des sommets conditionnels auxquels sont associées des conditions logiques $X = \{x_1, \ldots, x_L\}$. On demande de trouver, parmi tous les schémas-graphes équivalents à $\Gamma\,(Y_0, Y_{\text{fin}}, Y, P)$, celui qui présente le plus petit nombre de sommets conditionnels.

Montrons qu'on peut diviser le problème de minimisation du SGA $\Gamma\,(Y_0, Y_{\text{fin}}, Y, P)$ en plusieurs sous-problèmes indépendants en décomposant le SGA Γ en sous-graphes $\Gamma^1, \ldots, \Gamma^H$ tels qu'en associant à chacun d'eux un sous-graphe équivalent $\Gamma^h_{\min}$ $(h = 1, \ldots, H)$ ayant le plus petit nombre de sommets conditionnels, le graphe obtenu par réunion de ces derniers sous-graphes, noté $\Gamma_{\min}$, présente le plus petit nombre de sommets conditionnels.

La décomposition du SGA $\Gamma\,(Y_0, Y_{\text{fin}}, Y, P)$ sera faite de la façon suivante.

Etablissons-nous sur un opérateur quelconque dans l'ensemble $\{Y_0, Y_1, \ldots, Y_T\}$, par exemple sur Y_0, et cherchons les opérateurs qui admettent des chemins du type

$$Y_t p_{t1}^{e_{t1}} \ldots p_{tm}^{e_{tm}} \ldots p_{tM}^{e_{tM}} Y_s \tag{7-1}$$

passant uniquement par des sommets conditionnels; ici $\{p_{t1}, \ldots, p_{tM}\} \subseteq P$ est l'ensemble des sommets conditionnels traversés par le chemin de Y_t à Y_s; $e_{tm} \in \{0, 1\}$ est le symbole associé à la sortie du sommet conditionnel p_{tm} par laquelle passe le chemin considéré.

Sur le SGA Γ de la figure 7-1, ce seront les opérateurs Y_1, Y_3, Y_5. Sur la figure 7-2, *a* ils sont désignés par des flèches issues de l'opérateur Y_0. Ceci fait, cherchons les opérateurs à partir desquels il est possible d'arriver, par des chemins du type (7-1), à chacun des opérateurs de l'ensemble $\{Y_1, Y_3, Y_5\}$. Les opérateurs trouvés seront inscrits à gauche, au-dessous de Y_0. Ceci fait, on cherche, pour chacun des opérateurs situés à gauche, les opérateurs auxquels on peut aboutir par des chemins du type (7-1), et ainsi de suite, jusqu'à ce que les ensembles de gauche (A^1) et de droite (B^1) cessent de s'agrandir.

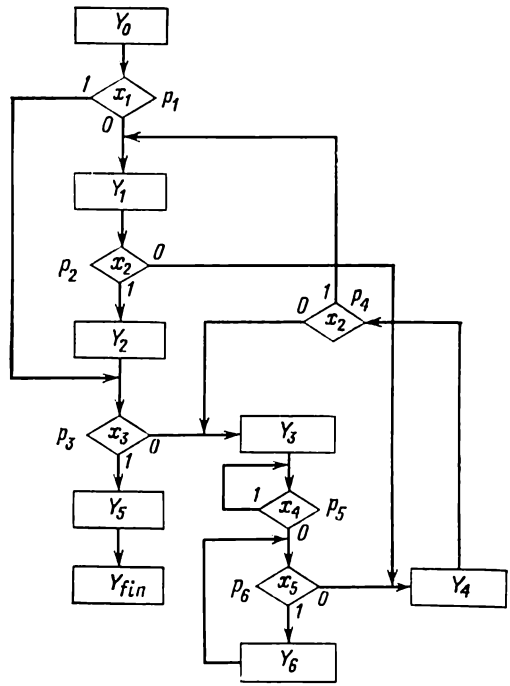


Fig. 7-1. Le SGA Γ avant la décomposition

Ensuite on s'établit sur un nouveau sommet $Y_m \notin A^1$ et l'on construit A^2 et B^2 comme décrit.

L'algorithme se déroule en H pas ($H \leqslant T + 1$, où T est le nombre des sommets opératoriels). A chaque h-ième pas on choisit un sommet $Y_{fin} \notin \bigcup_{i=1}^{h-1} A^i$, on construit A^h et B^h et ainsi de suite, jusqu'à ce que, au bout de H pas, il ne reste plus de sommets, utilisables comme points de départ, non compris dans les ensembles déjà construits $A^1, \ldots, A^H$.

On obtient en résultat une partition du schéma-graphe Γ en sous-graphes $\Gamma^1, \ldots, \Gamma^H$, où chaque sous-graphe Γ^h (A^h, B^h, P^h) est défini par les ensembles A^h, B^h, P^h et par les arcs incidents aux sommets réunis par les ensembles indiqués.

Du procédé de construction de la partition de Γ il ressort que

1) pour deux sous-graphes quelconques Γ^i et Γ^j on a $A^i \cap A^j = \varnothing$, $B^i \cap B^j = \varnothing$, $P^i \cap P^j = \varnothing$;

2) la partition est unique, sans rapport avec le choix du sommet de départ pour la construction des sous-graphes à chaque pas.

De l'unicité de la partition et de la condition $P^i \cap P^j = \varnothing$ on tire immédiatement la conclusion suivante. Soient R le nombre de

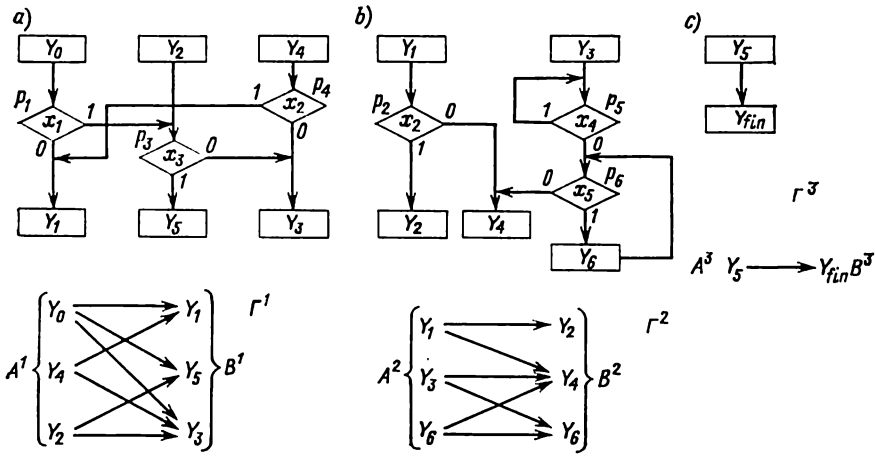


Fig. 7-2. Sous-graphes et leur représentation conventionnelle: Γ^1 (*a*), Γ^2 (*b*), Γ^3 (*c*)

sommets conditionnels de Γ et R^h le nombre de sommets conditionnels dans Γ^h ($h = 1, \ldots, H$). Alors $R = \sum_{h=1}^{H} R^h$.

Ceci établi, on montre sans peine que si Γ admet une partition en sous-graphes $\Gamma^1, \ldots, \Gamma^H$ où chaque sous-graphe Γ^h ($h = 1, \ldots, H$) contient un nombre minimal de sommets conditionnels $R^h_{\min}$, le graphe Γ contient lui aussi un nombre minimal de sommets conditionnels, égal à $\sum_{h=1}^{H} R^h_{\min}$.

On voit sur la figure 7-2, *a*, *b*, *c* une représentation conventionnelle du processus de partition du SGA Γ et les sous-graphes qui en résultent. Dans ces sous-graphes

$$\Gamma^1\colon A^1 = \{Y_0, Y_2, Y_4\},\ B^1 = \{Y_1, Y_3, Y_5\},\ P^1 = \{p_1, p_3, p_4\};$$
$$\Gamma^2\colon A^2 = \{Y_1, Y_3, Y_6\},\ B^2 = \{Y_2, Y_4, Y_6\},\ P^2 = \{p_2, p_5, p_6\};$$
$$\Gamma^3\colon A^3 = \{Y_5\},\ B^3 = \{Y_{fin}\},\ P^3 = \varnothing.$$

Les numéros des sommets conditionnels du SGA Γ et des sous-graphes sont indiqués à côté des sommets. Il est curieux qu'on a

$A^2 \cap B^2 = \{Y_6\} \neq \varnothing$; cela est admissible (on admet que l'intersection $A^i \cap B^i$ soit non vide ; au contraire, s'il y a $A^i \cap A^j = \varnothing$, il doit y avoir toujours également $B^i \cap B^j = \varnothing$).

R. Miller [45] décrit en détail les algorithmes de minimisation des circuits de commutation proposés par J. Roth. Ils sont fondés sur une *représentation géométrique* de la fonction booléenne réalisant une application de l'espace cartésien Z_2^n dans l'espace Z_2 :

$$Z_2^n \xrightarrow{f} Z_2,$$

où $Z_2 = \{0, 1\}$, $Z_2^n = \underbrace{Z_2 \times Z_2 \times \ldots \times Z_2}_{n \text{ fois}}$, $f^{-1}(1)$ est l'image réciproque de l'unité, et $f^{-1}(0)$, l'image réciproque du zéro.

Les éléments $f^{-1}(1)$ sont appelés 0-cubes ou sommets. Deux 0-cubes forment un 1-cube s'ils ne diffèrent que par les valeurs d'une seule composante (coordonnée). Dans ce cas les 0-cubes sont appelés faces du 1-cube. Par exemple, les 0-cubes 0011 et 0111 forment un 1-cube $0x11$, où la deuxième composante est dite libre, tandis que les autres sont liées. Deux 1-cubes forment un 2-cube s'ils admettent la même composante libre et s'ils ne diffèrent que par la valeur d'une seule composante liée. C'est ainsi que les 1-cubes $0x11$ et $0x10$ forment un 2-cube $0x1x$. Par récurrence, on introduit la notion de r-cube ($r \leqslant n$). A la fonction booléenne qui s'écrit analytiquement sous la forme

$$f = \bar{x}_1\bar{x}_3 \vee x_1\bar{x}_2\bar{x}_4 \vee x_2x_3\bar{x}_4$$

correspond un ensemble de cubes (recouvrement cubique)

$$K(f) = \begin{Bmatrix} 0 & x & 0 & x \\ 1 & 0 & x & 0 \\ x & 1 & 1 & 0 \end{Bmatrix}.$$

Par analogie à la représentation géométrique des fonctions booléennes, introduisons une représentation géométrique des formules de transition. Soit $Y_f \rightarrow \bigvee_{t=1}^{T+1} \alpha_{ft} Y_t$ une formule de transition. Mettons la fonction de transition $\alpha_{ft}(x_1, \ldots, x_n)$ de Y_f à Y_t ($t = 1, \ldots, T+1$) sous forme d'un recouvrement cubique et associons à chacun des cubes du recouvrement son opérateur Y_t ; un cube n-dimensionnel auquel on a associé un opérateur sera appelé *cube marqué*. Si D_t est l'ensemble des cubes marqués engendrés par la fonction α_{ft}, à la formule de transition $Y_f \rightarrow \bigvee_{t=1}^{T+1} \alpha_{ft} Y_t$ correspond un ensemble de cubes $D(Y_f) = \bigcup_{t=1}^{T+1} D_t$, auquel nous donnerons le nom de *recouvrement de la formule de transition*. C'est ainsi qu'à la

formule de transition de l'opérateur Y_2 sur la figure 7-3

$$Y_2 \to (\bar{x}_1 x_2 \vee x_1 \bar{x}_2 \bar{x}_3)\, Y_4 \vee (\bar{x}_1 \bar{x}_2 x_3 \vee \bar{x}_1 \bar{x}_2 \bar{x}_3)\, Y_5 \vee$$
$$\vee\, x_1 x_2 Y_6 \vee x_1 \bar{x}_2 x_3 Y_7$$

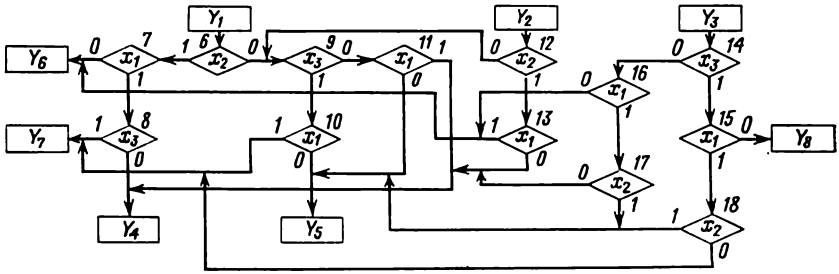


Fig. 7-3. Sous-graphe à trois sommets de départ

correspond un recouvrement $D\,(Y_2)$ qui s'écrit

$$D\,(Y_2) = \left\{\begin{matrix} 0 & 1 & x & Y_4 \\ 1 & 0 & 0 & Y_4 \\ 0 & 0 & 1 & Y_5 \\ 0 & 0 & 0 & Y_5 \\ 1 & 1 & x & Y_6 \\ 1 & 0 & 1 & Y_7 \end{matrix}\right\}. \tag{7-2}$$

Nous représenterons un tel recouvrement de la façon suivante:

$$D\,(Y_2) = \left|\begin{matrix} 0 & 1 & x & 4 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 5 \\ 1 & 1 & x & 6 \\ 1 & 0 & 1 & 7 \end{matrix}\right|.$$

Introduisons une opération $\times$ qui sera effectuée sur un couple de cubes marqués $c^r Y_l = (a_1, \ldots, a_i, \ldots, a_n)\, Y_l$ et $c^s Y_m = (b_1, \ldots, b_i, \ldots, b_n)\, Y_m$ appartenant au recouvrement de la formule de transition

$$Y_f \to \bigvee_{t=1}^{T+1} \alpha_{ft} Y_t \;\; (Y_l,\, Y_m \in \{Y_1, \ldots, Y_{T+1}\});$$

l'opération $\times$ conduit à un nouveau cube $c^p Y_q = c^r Y_l \times c^s Y_m$ (on a $Y_q \notin \{Y_0, Y_1, \ldots, Y_{T+1}\}$ si $l \neq m$). L'opération se fait séparément sur les parties logiques ($c^r \times c^s$) et sur les parties opé-

ratorielles $(Y_l \times Y_m)$ des cubes marqués. Puisque l'opération $c^r \times \times c^s$ se fait sur des cubes à n coordonnées, nous définirons l'opéra-

Tableau 7-1

Table de l'opération $\times$

$\times$	0	1	x
0	0	y	0
1	y	1	1
x	0	1	x

tion du produit de coordonnées à l'aide du tableau 7-1 :

$$c^p = (a_1, \ldots, a_i, \ldots, a_n) \times (b_1, \ldots, b_i, \ldots, b_n),$$

$$c^p = \begin{cases} (m\,(a_1 \times b_1), \ldots, m\,(a_i \times b_i), \ldots, m\,(a_n \times b_n)) & \text{si } y \text{ intervient exactement en une coordonnée ; } m\,(a_i \times b_i) \text{ est } m\,(0) = 0,\ m\,(1) = 1,\ m\,(x) = m\,(y) = x\,; \\ \varnothing & \text{dans le cas contraire (c'est-à-dire si } y \text{ n'intervient en aucune coordonnée ou intervient en deux coordonnées ou plus).} \end{cases}$$

Géométriquement, l'opération $c^r \times c^s$ donne le plus grand cube c^p qui vérifie les conditions suivantes :

— ses faces opposées sont les cubes c^r et c^s ;

— aucun des cubes initiaux ne recouvre à lui seul le cube c^p.

C'est précisément cette dernière condition qui distingue l'opération $\times$ de l'opération du $*$-produit proposé par J. Roth (voir [45]) :

$$\text{a)}\ \left.\begin{array}{l} (1xx) \times (011) = (x11) \\ (1xx) * (011) = (x11) \end{array}\right\} \text{ même résultat,}$$

$$\text{b)}\ \left.\begin{array}{l} (0xx) \times (xx1) = \varnothing, \\ (0xx) * (xx1) = (0x1) \end{array}\right\} \text{ résultats différents.}$$

L'opération $\times$ sur les parties opératorielles dépend du résultat de l'opération effectuée sur les parties logiques. Supposons que $c^r \times c^s = c^p \neq \varnothing$ et que les cubes c^r, c^s admettent dans la coordonnée i des valeurs opposées $(a_i = e_i,\ b_i = \bar{e}_i,\ e_i \in \{0, 1\})$. On fait correspondre alors à l'opérateur Y_q marquant le cube c^p la formule de transition

$$Y_q \rightarrow x_i^{e_i} Y_l \vee x_i^{\bar{e}_i} Y_m \tag{7-3}$$

dans laquelle $x_i^1 = x_i$, $x_i^0 = \bar{x}_i$.

En notation abrégée, nous écrirons la formule de transition comme suit :

$$Y_q = x_i Y_l Y_m. \tag{7-4}$$

Dans cette notation, x_i est suivi par l'opérateur auquel on passe dans (7-3) à partir de Y_q si $x_i = 1$, puis vient l'opérateur auquel on passe si $x_i = 0$. De toute évidence, on suppose en écrivant (7-4) que $a_i = e_i = 1$, $b_i = \bar{e}_i = 0$. Si $a_i = e_i = 0$, $b_i = \bar{e}_i = 1$, on devra permuter Y_l et Y_m : $Y_q = x_i Y_m Y_l$. Si $Y_l = Y_m$, on tire de (7-3)

$$Y_q \to x_i^{e_i} Y_m \vee x_i^{\bar{e}_i} Y_m = Y_m.$$

On a donc

$$c^r Y_l \times c^s Y_m = \begin{cases} \varnothing \text{ si } c^r \times c^s = \varnothing; \\ c^p Y_q \text{ si } c^r \times c^s \neq \varnothing; \text{ ici } c^p = c^r \times c^s \\ \quad \text{et } Y_q \text{ se définit par (7-3)} \end{cases}$$

Remarquons qu'à la formule de transition (7-3), ou à sa notation abrégée (7-4) que nous utiliserons dans le texte qui suit, correspond le sous-graphe montré sur la figure 7-4. L'opérateur Y_q nouvellement introduit sera appelé opérateur *dérivé* et représenté sur le SGA par un rond, afin de le distinguer des opérateurs initiaux (primitifs) $Y_0, Y_1, \ldots, Y_{T+1}$.

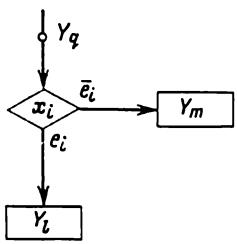


Fig. 7-4. Sous-graphe correspondant à la formule de transition $Y_q \to x_i^{e_i} Y_l \vee \vee x_i^{\bar{e}_i} Y_m$

L'opération $\times$ introduite sur un couple de cubes marqués $c^r Y_l$, $c^s Y_m$ sera appelée *opération de recollement* : nous dirons que ces cubes sont recollés suivant la variable x_i s'ils admettent des valeurs opposées (0 et 1) dans la coordonnée i.

Introduisons ensuite l'opération de *recollement complet* : elle se fait sur deux cubes marqués de la même façon qui ont toutes leurs coordonnées identiques à l'exception de celle qui sert au recollement. Voici un exemple de recollement complet :

$$\begin{array}{r} \times \begin{array}{l} 1\ x\ 0\ x\ 1\ Y_8 \\ 1\ x\ 1\ x\ 1\ Y_8 \end{array} \\ \hline 1\ x\ x\ x\ 1\ Y_8. \end{array}$$

Par contre,

$$\begin{array}{r} \times \begin{array}{l} 1\ x\ 0\ x\ 1\ Y_8 \\ 1\ x\ 1\ 1\ x\ Y_8 \end{array} \\ \hline 1\ x\ x\ 1\ 1\ Y_8 \end{array}$$

n'est pas un recollement complet, car les cubes choisis pour l'opération diffèrent dans plus d'une coordonnée. De même,

$$\begin{array}{r} \times \begin{array}{l} 1\ x\ 0\ x\ 1\ Y_8 \\ 1\ x\ 1\ x\ 1\ Y_9 \end{array} \\ \hline 1\ x\ x\ x\ 1\ Y_{10} \end{array}, \quad Y_{10} = x_3 Y_9 Y_8,$$

n'est pas un recollement complet, vu que les cubes opérandes, bien que ne différant que d'une seule coordonnée, sont marqués par des opérateurs différents.

Nous dirons que deux cubes marqués $c^r Y_l = (a_1, \ldots, a_n)\, Y_l$ et $c^s Y_m = (b_1, \ldots, b_n)\, Y_m$ sont deux faces opposées du cube marqué $c^p Y_q$ si c^r et c^s sont deux faces opposées du cube c^p (ont toutes leurs coordonnées identiques sauf une, par exemple i, en laquelle ils prennent des valeurs opposées), et $Y_q = x_i Y_l Y_m$ si $a_i = 1$, $b_i = 0$ et $Y_q = x_i Y_m Y_l$ si $a_i = 0$, $b_i = 1$.

Introduisons la relation de recouvrement pour un couple de cubes identiquement marqués $c^r Y_m$ et $c^p Y_m$. Nous dirons que $c^r Y_m$ *recouvre* $c^p Y_m$, ou que $c^p Y_m$ admet $c^r Y_m$ comme recouvrement ($c^p Y_m \subseteq c^r Y_m$), si c^r recouvre c^p ($c^p \subseteq c^r$) (voir [45]), c'est-à-dire si c^p est une face de c^r. Il est évident que la relation de recouvrement est réflexive, transitive et antisymétrique.

Nous décrirons maintenant un algorithme de recherche, dans le sein d'un SGA, d'un ensemble de sous-graphes équivalents parmi lesquels sera trouvé, en réalisant un recouvrement d'un type particulier, le sous-graphe qui possède le plus petit nombre de sommets conditionnels. Convenons d'opérer dans une classe de schémas-graphes où un chemin du type (7-1) ne passe jamais par deux sommets conditionnels contenant une même condition logique. La nécessité d'une pareille restriction est évidente: à chaque chemin du type (7-1) on associe un cube marqué dont chaque coordonnée correspond à une condition logique.

Supposons que la partition du SGA nous a donné un sous-graphe Γ contenant m sommets opératoriels de départ $Y_1, \ldots, Y_m$ et n conditions logiques. Il est évident qu'on peut construire comme précédemment pour chaque opérateur de départ Y_j un recouvrement marqué D^j ($j = 1, \ldots, m$). Pour le sous-graphe Γ à trois sommets opératoriels de départ représenté sur la figure 7-3, les trois recouvrements se présentent comme suit:

$$D^1 = \left|\begin{array}{ccc|c} 1 & 0 & 0 & 4 \\ 1 & 1 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 5 \\ 0 & 1 & x & 6 \\ 1 & 1 & 1 & 7 \\ 1 & 0 & 1 & 7 \end{array}\right|, \quad D^2 = \left|\begin{array}{ccc|c} 0 & 1 & x & 4 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 5 \\ 1 & 1 & & 6 \\ 1 & 0 & 1 & 7 \end{array}\right|, \quad D^3 = \left|\begin{array}{ccc|c} 1 & 0 & 0 & 4 \\ 1 & 1 & 1 & 5 \\ 1 & 1 & 0 & 5 \\ 0 & x & 0 & 6 \\ 1 & 0 & 1 & 7 \\ 0 & x & 1 & 8 \end{array}\right|.$$

Faisons, dans chaque recouvrement, toutes les opérations possibles de recollement complet et d'*absorption*. L'opération d'absorption consiste à éliminer tous les cubes qui sont recouverts par d'autres cubes. Dans notre exemple

$$D_0^1 = \left|\begin{array}{ccc|c} 1 & x & 0 & 4 \\ 0 & 0 & x & 5 \\ 0 & 1 & x & 6 \\ 1 & x & 1 & 7 \end{array}\right|, \quad D_0^2 = \left|\begin{array}{ccc|c} 0 & 1 & x & 4 \\ 1 & 0 & 0 & 4 \\ 0 & 0 & x & 5 \\ 1 & 1 & x & 6 \\ 1 & 0 & 1 & 7 \end{array}\right|, \quad D_0^3 = \left|\begin{array}{ccc|c} 1 & 0 & 0 & 4 \\ 1 & 1 & x & 5 \\ 0 & x & 0 & 6 \\ 1 & 0 & 1 & 7 \\ 0 & x & 1 & 8 \end{array}\right|. \tag{7-5}$$

Le zéro en indice inférieur de D_0^j indique que c'est un recouvrement initial ($j = 1, \ldots, m$). Les opérateurs qui marquent les cubes des D_0^j seront appelés *primaires*. L'ensemble des opérateurs primaires des D_0^j sera noté G_0^j. Dans notre exemple, $G_0^1 = G_0^2 = \{Y_4, Y_5, Y_6, Y_7\}$, $G_0^3 = \{Y_4, Y_5, Y_6, Y_7, Y_8\}$, $G_0 = G_0^1 \cup G_0^2 \cup G_0^3 = \{Y_4, Y_5, Y_6, Y_7, Y_8\}$.

L'algorithme de recherche des sous-graphes équivalents d'un SGA se déroule en H pas ($H \leqslant n$, où n est le nombre des conditions logiques du sous-graphe).

En premier pas, nous appliquerons l'opération $\times$ à chaque couple de cubes $(c^r Y_{l_0}, c^s Y_{m_0})$ des D_0^j. Elle nous donnera des ensembles de cubes $\widetilde{D}_1^j$ ($j = 1, \ldots, m$) *):

$$\widetilde{D}_1^1 = \left|\begin{array}{ccc|c} \underline{x} & 0 & 0 & 10 \\ \underline{x} & 1 & 0 & 11 \\ 1 & x & \underline{x} & 12 \\ 0 & \underline{x} & x & 13 \\ \underline{x} & 0 & 1 & 14 \\ \underline{x} & 1 & 1 & 15 \end{array}\right|, \quad \widetilde{D}_1^2 = \left|\begin{array}{ccc|c} 0 & \underline{x} & x & 16 \\ \underline{x} & 1 & x & 17 \\ \underline{x} & 0 & 0 & 10 \\ 1 & \underline{x} & 0 & 18 \\ 1 & 0 & \underline{x} & 12 \\ \underline{x} & 0 & 1 & 14 \\ 1 & \underline{x} & 1 & 19 \end{array}\right|, \quad \widetilde{D}_1^3 = \left|\begin{array}{ccc|c} 1 & \underline{x} & 0 & 20 \\ \underline{x} & 0 & 0 & 11 \\ 1 & 0 & \underline{x} & 12 \\ \underline{x} & 1 & 0 & 21 \\ 1 & \underline{x} & 1 & 22 \\ \underline{x} & 1 & 1 & 23 \\ 0 & x & \underline{x} & 24 \\ \underline{x} & 0 & 1 & 25 \end{array}\right|.$$

Inscrivons les opérateurs dérivés ($Y_{q_1} = x_i Y_l Y_{\upsilon m_0}$) dans une table séparée (tableau 7-2) en mettant dans la première colonne l'indice q_1, dans la deuxième la variable de recollement x_i, dans la troisième et la quatrième les indices l_0 et m_0 des opérateurs respectivement :

$$q_1 \mid x_i \mid l_0 \mid m_0.$$

---

*) Dans cet exemple les opérateurs dérivés (distincts des opérateurs primaires) sont numérotés à partir du nombre 10.

Tableau 7-2

Fonction f_1

10	x_1	4	5
11	x_1	4	6
12	x_3	7	4
13	x_2	6	5
14	x_1	7	5
15	x_1	7	6

16	x_2	4	5
17	x_1	6	4
18	x_2	6	4
19	x_2	6	7
20	x_2	5	4

21	x_1	5	6
22	x_2	5	7
23	x_1	5	8
24	x_3	8	6
25	x_1	7	8

Dans chacun des cubes des $\widetilde{D}_1^j$ soulignons la coordonnée suivant laquelle on a effectué le recollement qui a fait naître le cube en question. Par exemple,

$$\begin{array}{r} \times \begin{array}{r} 1\ x\ 0\ Y_4 \\ 1\ x\ 1\ Y_7 \end{array} \\ \hline 1\ x\ \underline{x}\ Y_{12} \end{array}; \quad Y_{12} = x_3 Y_7 Y_4,$$

ou, ce qui revient au même,

$$1\ x\ \underline{x}\ |\ 12; \quad 12\ |\ x_3\ |\ 7\ |\ 4.$$

Ajoutons à l'ensemble $\widetilde{D}_1^j$ les cubes $E_0^j \subseteq D_0^j$ dont le rang est non inférieur à l'unité. Dans notre exemple

$$E_0^1 = \left|\begin{array}{ccc|c} 1 & x & 0 & 4 \\ 0 & 0 & x & 5 \\ 0 & 1 & x & 6 \\ 1 & x & 1 & 7 \end{array}\right|, \quad E_0^2 = \left|\begin{array}{ccc|c} 0 & 1 & x & 4 \\ 0 & 0 & x & 5 \\ 1 & 1 & x & 6 \end{array}\right|, \quad E_0^3 = \left|\begin{array}{ccc|c} 1 & 1 & x & 5 \\ 0 & x & 0 & 6 \\ 0 & x & 1 & 8 \end{array}\right|.$$

Construisons les ensembles $D_1^j = \widetilde{D}_1^j \cup E_0^j$:

$$D_1^1 = \left|\begin{array}{ccc|c} \underline{x} & 0 & 0 & 10 \\ \underline{x} & 1 & 0 & 11 \\ 1 & x & \underline{x} & 12 \\ 0 & \underline{x} & x & 13 \\ \underline{x} & 0 & 1 & 14 \\ \underline{x} & 1 & 1 & 15 \\ 1 & x & 0 & 4 \\ 0 & 0 & x & 5 \\ 0 & 1 & x & 6 \\ 1 & x & 1 & 7 \end{array}\right|, \quad D_1^2 = \left|\begin{array}{ccc|c} 0 & \underline{x} & x & 16 \\ \underline{x} & 1 & x & 17 \\ \underline{x} & 0 & 0 & 10 \\ 1 & \underline{x} & 0 & 18 \\ 1 & 0 & \underline{x} & 12 \\ \underline{x} & 0 & 1 & 14 \\ 1 & \underline{x} & 1 & 19 \\ 0 & 1 & x & 4 \\ 0 & 0 & x & 5 \\ 1 & 1 & x & 6 \end{array}\right|, \quad D_1^3 = \left|\begin{array}{ccc|c} 1 & \underline{x} & 0 & 20 \\ \underline{x} & 0 & 0 & 11 \\ 1 & 0 & \underline{x} & 12 \\ \underline{x} & 1 & 0 & 21 \\ 1 & \underline{x} & 1 & 22 \\ \underline{x} & 1 & 1 & 23 \\ 0 & x & \underline{x} & 24 \\ \underline{x} & 0 & 1 & 25 \\ 1 & 1 & x & 5 \\ 0 & x & 0 & 6 \\ 0 & x & 1 & 8 \end{array}\right|,$$

Le tableau 7-2 donne une fonction f_1 qui est définie dans l'ensemble de triplets (x_i, Y_{l_0}, Y_{m_0}) et qui prend une valeur dans l'en-

semble des opérateurs dérivés $\widetilde{G}_1 = \bigcup_{j=1}^{m} \widetilde{G}_1^j$, où les opérateurs de $\widetilde{G}_1^j$ marquent les cubes de $\widetilde{D}_1^j$ ($x_i \in X$; Y_{l_0}, $Y_{m_0} \in G_0$). A chaque élément $Y_{q_1} \in \widetilde{G}_1$ correspond un sous-graphe de SGA analogue à celui de la figure 7-4 et possédant un sommet conditionnel unique.

Tableau 7-3

Table de l'opération $\times$ à partir du pas n°2

$\times$	0	1	x	$\underline{x}$
0	0	y	0	$\varnothing$
1	y	1	1	$\varnothing$
x	0	1	x	$\underline{x}$
$\underline{x}$	$\varnothing$	$\varnothing$	$\underline{x}$	$\underline{x}$

Si P_0^j et G_1^j sont des ensembles d'opérateurs marquant les cubes dans les recouvrements E_0^j et D_1^j respectivement, on a $G_1^j = \widetilde{G}_1^j \cup \cup P_0^j$, $G_1 = \bigcup_{j=1}^{m} G_1^j$.

Remarque. Chacun des h pas suivants ($h = 2, \ldots, H$) commencera par l'opération $D_{h-1}^j \times D_{h-1}^j$. Cette opération sera cependant interdite sur des couples de cubes $c^r Y_{l_{h-1}}$, $c^s Y_{m_{h-1}}$ de D_{h-1}^j dont l'i-ième coordonnée est égale à x souligné ($\underline{x}$) dans un cube et à 0 ou 1 (coordonnée liée) dans l'autre. C'est ainsi que, sur le deuxième pas, l'opération $\times$ ne se fera pas sur le couple de cubes $11\underline{x} \mid 6$ et $0\underline{x}x \mid 16$. De toute évidence, cette restriction nous renvoie à une

Tableau 7-4

Fonction f_2

26	x_2	11	10
27	x_3	14	10
28	x_3	15	11
29	x_1	12	13
30	x_1	12	5
31	x_1	12	6
32	x_1	4	13
33	x_1	7	13
34	x_2	15	14

35	x_1	18	16
36	x_1	19	16
37	x_2	17	10
38	x_2	17	14
39	x_3	19	18
40	x_2	6	12
41	x_3	22	20
42	x_1	20	6

43	x_2	21	11
44	x_3	25	11
45	x_1	12	24
46	x_2	5	12
47	x_3	23	21
48	x_1	22	8
49	x_2	23	25
50	x_1	5	24

classe de schémas-graphes où aucun chemin joignant deux sommets opératoriels ne passe par deux ou plusieurs sommets conditionnels de même condition logique ; nous avons déjà rencontré cette classe plus haut. Ainsi donc, sur les pas n°s 2, . . ., H, l'opération $c^p = (a_1, \ldots, a_n) \times (b_1, \ldots, b_n)$ se définit, conformément au ta-

bleau 7-3, comme suit :

$$c^p = \begin{cases} (m\,(a_1 \times b_1),\ \ldots,\ m\,(a_i \times b_i),\ \ldots,\ m\,(a_n \times b_n)) \\ \quad \text{si } y \text{ intervient exactement en une coordonnée et si aucune} \\ \quad \text{coordonnée n'est égale à } \varnothing \text{ ; ici } m\,(a_i \times b_i) \text{ est } m\,(0) = 0, \\ \quad m\,(1) = 1,\ m\,(x) = x,\ m\,(y) = m\,(\underline{x}) = \underline{x}; \\ \varnothing \text{ dans le cas contraire.} \end{cases}$$

Faisant sur le pas n° 2 l'opération $D_1^j \times D_1^j$, on obtient : $\widetilde{D}_2^j = D_1^j \times D_1^j$, $D_2^j = \widetilde{D}_2^j \cup E_1^j$ (E_1^j est un ensemble de cubes de D_1^j de rang non inférieur à 2) ; $G_2^j = \widetilde{G}_2^j \cup P_1^j$ et le tableau d'une fonction f_2 (tableau 7-4) définie sur l'ensemble de triplets (x_i, Y_{l_1}, Y_{m_1}) et prenant une valeur dans l'ensemble $\widetilde{G}_2 = \bigcup_{j=1}^{m} \widetilde{G}_2^j$. Faisant toutes les absorptions possibles dans D_2^j, on obtient dans l'exemple considéré *) :

$$D_2^1 = \left|\begin{array}{ccc|l} \underline{x} & \underline{x} & 0 & 26, 32 \\ \underline{x} & 0 & \underline{x} & 27, 30 \\ \underline{x} & 1 & \underline{x} & 28, 31 \\ \underline{x} & \underline{x} & x & 29 \\ \underline{x} & \underline{x} & 1 & 33, 34 \\ 0 & \underline{x} & x & 13 \\ 1 & x & \underline{x} & 12 \end{array}\right| ; \quad D_2^2 = \left|\begin{array}{ccc|l} \underline{x} & \underline{x} & 0 & 35, 37 \\ \underline{x} & \underline{x} & 1 & 36, 38 \\ \underline{x} & 0 & \underline{x} & 27, 30 \\ 1 & \underline{x} & \underline{x} & 39, 40 \\ 0 & \underline{x} & x & 16 \\ \underline{x} & 1 & x & 17 \end{array}\right| ; \quad D_2^3 = \left|\begin{array}{ccc|l} 1 & \underline{x} & \underline{x} & 41, 46 \\ \underline{x} & \underline{x} & 0 & 42, 43 \\ \underline{x} & 0 & \underline{x} & 44, 45 \\ \underline{x} & 1 & \underline{x} & 47, 50 \\ \underline{x} & \underline{x} & 1 & 48, 49 \\ 0 & x & \underline{x} & 24 \end{array}\right| .$$

Procédant de la manière décrite, on obtient, sur un pas n° h ($h = 1, \ldots, H$; $H \leqslant n$), D_h^j, G_h^j et le tableau d'une fonction f_h définie sur l'ensemble des triplets $(x_i, Y_{l_{h-1}}, Y_{m_{h-1}})$ et prenant une valeur dans l'ensemble $\widetilde{G}_h = \bigcup_{j=1}^{m} \widetilde{G}_h^j$.

De toute évidence, au bout de H pas ($H \leqslant n$) l'ensemble D_H^j ne contiendra que des n-cubes marqués ($xx \ldots x$). Il est clair que le nombre des pas de l'algorithme n'est pas supérieur à n.

Reprenons notre exemple. Sur le troisième (et le dernier) pas nous obtenons

$$D_3^1 = |\,\underline{x}\underline{x}\underline{x}\,|\ 29,\ 51,\ 52,\ 53,\ 54,\ 55,\ 56,\ 57,\ 58;$$

$$D_3^2 = |\,\underline{x}\underline{x}\underline{x}\,|\ 59,\ 60,\ 61,\ 62,\ 63,\ 64,\ 65,\ 66:$$

$$D_3^3 = |\,\underline{x}\underline{x}\underline{x}\,|\ 67,\ 68,\ 69,\ 70,\ 71,\ 72,\ 73,\ 74,\ 75,\ 76.$$

---

*) Dans D_2^j les cubes $c^s Y_{m_r}$ et $c^r Y_{l_r}$ s'écrivent pour $c^r = c^s$ sous la forme $c^r | m_r,\ l_r$.

Après la dernier pas — le H-ième — nous obtenons des recouvrements D_H^j ($j = 1, \ldots, m$) dont les cubes sont marqués par des opérateurs provenant des ensembles G_H^j. Soit $G_H = \bigcup_{j=1}^{m} G_H^j$, où G_H^j est un sous-ensemble de G_H qui correspond à l'opérateur de départ Y_j. Dans notre exemple

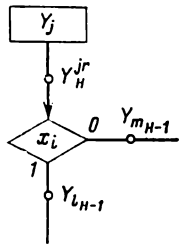


Fig. 7-5. Les opérateurs Y_j et Y_H^{jr} se confondent

$$G_H^1 = \{Y_{29}, Y_{51}, Y_{52}, \ldots, Y_{58}\},$$
$$G_H^2 = \{Y_{59}, Y_{60}, \ldots, Y_{66}\},$$
$$G_H^3 = \{Y_{67}, Y_{68}, \ldots, Y_{76}\}.$$

Soit $G_H^j = \{Y_H^{j1}, \ldots, Y_H^{jr}, \ldots, Y_H^{jR_j}\}$.

Puisque D_H^j ne contient que des cubes de dimension n ($xx \ldots x$), il est évident que le chemin allant de Y_j aux opérateurs de G_H^j traverse un ensemble vide de sommets conditionnels, ce qui revient à dire que Y_j se confond avec chacun des opérateurs de G_H^j. En mettant l'opérateur Y_H^{jr} sous la forme $x_i Y_{l_{H-1}}$, $Y_{m_{H-1}}$ (fonction f_H), nous obtenons un sous-graphe représenté sur la figure 7-5. En développant $Y_{l_{H-1}}$, $Y_{m_{H-1}}$, etc. à l'aide des fonctions $f_{H-1}, f_{H-2}, \ldots, f_1$, nous obtiendrons, au bout d'au plus H pas, les opérateurs primaires de G_0. Cela nous donnera une variante de réalisation de la portion de sous-graphe déterminée par le recouvrement D^j. Puisque G_H^j contient R_j éléments, il suffit de faire la procédure décrite R_j fois pour chaque $Y_H^{j1}, \ldots, Y_H^{jR_j}$ pour obtenir R_j sous-graphes équivalents pour le recouvrement D_j. Par exemple, en commençant par Y_{53} (tableau 7-5), on aboutit au sous-graphe montré sur la figure 7-6. Puisque G_H^1 admet

Tableau 7-5

Fonction f_3

51	x_3	33	26
52	x_3	34	26
53	x_3	33	32
54	x_3	34	32
55	x_2	28	27
56	x_2	28	30
57	x_2	31	27
58	x_2	31	30
59	x_3	36	35

60	x_3	36	37
61	x_3	38	35
62	x_3	38	37
63	x_2	17	27
64	x_2	17	30
65	x_1	39	16
66	x_1	40	16
67	x_1	41	24
68	x_1	46	24

69	x_3	48	42
70	x_3	48	43
71	x_3	49	42
72	x_3	49	43
73	x_2	47	44
74	x_2	47	45
75	x_2	50	44
76	x_2	50	45

neuf opérateurs dérivés, il existe neuf variantes possibles de réalisation de Y_1. De même, Y_2 et Y_3 admettent huit et dix réalisations respectivement. Toutes les variantes de réalisation sont résumées dans le tableau 7-6 dont les colonnes renferment les opérateurs développés par la mise en facteur des fonctions f_3, f_2 et f_1. Dans ce tableau les opérateurs dérivés sont mis entre parenthèses chaque fois qu'ils interviennent au moins une fois pour un $Y_f \neq Y_j$. Les opérateurs primaires ne figurent pas dans le tableau. En choisissant une variante pour chacun des Y_1, Y_2, Y_3, on obtient évidemment un sous-graphe de SGA équivalent au sous-graphe Γ de la figure 7-3; il contiendra autant de sommets conditionnels qu'il y a d'opérateurs dérivés (distincts) dans les variantes choisies. Par exemple, en choisissant les variantes b_7, b_{17} et b_{19}, on obtient un SGA à douze sommets conditionnels.

Supposons qu'il y a m opérateurs de départ $Y_1, \ldots, Y_m$; le tableau des variantes prévoit R_j variantes de réalisation ($j = 1, \ldots, m$) pour chacun. Pour trouver un sous-graphe possédant le plus petit nombre de sommets conditionnels, il suffit de choisir dans le tableau une variante pour chaque Y_j de telle façon que le nombre total des opérateurs dérivés distincts dans ces variantes soit minimal. C'est le problème de recherche d'un recouvrement minimal du tableau des variantes. On comprend que le nombre total des sous-

Tableau 7-6

Table des variantes de réalisation de D^1, D^2 et D^3

Y_1									Y_2									Y_3									
b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}	b_{16}	b_{17}	b_{18}	b_{19}	b_{20}	b_{21}	b_{22}	b_{23}	b_{24}	b_{25}	b_{26}	b_{27}
29	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76
(12)	33	34	33	34	28	28	31	31	36	36	38	38	17	17	39	40	41	46	48	48	49	49	47	47	50	50
13	26	26	32	32	(27)	(30)	(27)	(30)	35	37	35	37	(27)	(30)	16	16	24	24	42	43	42	43	44	45	44	45
	13	15	13	15	15	15	12	(12)	19	19	17	17	(14)	(12)	19	(12)	22	(12)	22	22	23	23	23	23	24	24
		(11)	(14)	(14)	(11)	(11)	(14)		16	16	(14)	(14)	(10)		18		20		20	21	25	25	21	21	25	(12)
		(10)	(11)	13	(14)	(12)	(10)		18	17	18	(10)								(11)	20	21	25	(12)	(11)	
			(10)		(10)					(10)	16											(11)	(11)	24		

graphes qu'on peut tracer d'après ce tableau est égal à $R_1 \times \ldots \times R_m$; dans l'exemple du tableau 7-6, on a $9 \cdot 8 \cdot 10 = 720$ sous-graphes distincts.

On pourrait chercher un recouvrement minimal du tableau des variantes en modifiant la *méthode de Petrick* par laquelle on cherche un recouvrement minimal du tableau de Quine lors de la minimisation d'un système de fonctions booléennes [22]. Supposons que le tableau des variantes propose pour chaque Y_j ($j = 1, \ldots, m$) des variantes $b_{j1}, \ldots, b_{jr}, \ldots, b_{jR_j}$. Chaque variante peut être mise sous forme de conjonction K_{jr} de symboles des opérateurs dérivés figurant dans la variante considérée. C'est ainsi qu'à la variante b_5 (tableau 7-6) correspond la conjonction $54 \wedge 34 \wedge 32 \wedge 15 \wedge 14 \wedge 13$. L'ensemble des variantes pour un opérateur Y_j peut s'écrire $\bigvee_{r=1}^{R_j} K_{jr}$; l'ensemble des variantes pour tous les opérateurs $Y_1, \ldots, Y_m$ s'écrira alors

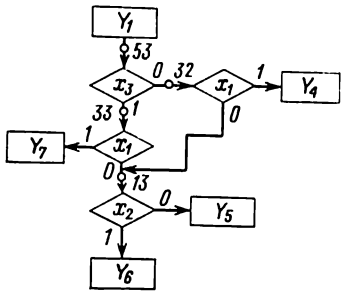


Fig. 7-6. Une variante de réalisation du sous-graphe avec opérateur de départ Y_1

$$\bigwedge_{j=1}^{m} \left(\bigvee_{r=1}^{R_j} K_{jr} \right).$$

Chassant les parenthèses dans la dernière expression et faisant toutes les opérations possibles du type $AA = A$, on obtient une disjonction de $R_1 \ldots R_m$ conjonctions, où R_j ($j = 1, \ldots, m$) est le nombre des variantes de réalisation de Y_j. En cherchant à partir de cette expression la conjonction qui possède le plus petit nombre de facteurs, on obtient un recouvrement minimal du tableau des variantes et, par conséquent, un sous-graphe minimal du SGA.

Même avec un ordinateur puissant, ce problème présente de grosses difficultés : l'augmentation du nombre d'opérateurs de départ et de variantes de réalisation des opérateurs entraîne une croissance très rapide du nombre des conjonctions obtenues. Soient en effet $m = 10$, $R_j = 10$; pour $j = 1, \ldots, m$ le nombre des conjonctions s'élève à 10^{10}.

Il existe un procédé par lequel on arrive à réduire sensiblement le tableau des variantes sans risque de perte de recouvrement minimal. Introduisons la relation de recouvrement entre variantes sur l'ensemble des variantes pour chaque Y_j ($j = 1, \ldots, m$) en disant que b_s *recouvre* b_t (où b_s et b_t sont deux variantes appartenant au bloc Y_j) si, quel que soit le choix des variantes dans tous les autres blocs $Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_m$, le nombre d'opérateurs dérivés ajouté à un quelconque recouvrement (y compris minimal) de

tout le tableau dans le cas de la variante b_s n'est pas supérieur au nombre d'opérateurs dérivés qu'apporte avec elle la variante b_t (ce cas se note $b_s \leqslant b_t$). Il est clair que si $b_s \leqslant b_t$, on peut éliminer la variante b_t du tableau des variantes sans risquer de manquer de recouvrement minimal.

Il est facile de montrer qu'on a toujours $b_s \leqslant b_t$ si les deux conditions suivantes sont vérifiées:

1) $L_s \leqslant L_t$,

2) $(L_t - L_s) - P(B_t \setminus B_s) \geqslant 0$ *).

Dans ces expressions:

L_s et L_t sont les longueurs des variantes b_s et b_t (le nombre d'opérateurs dérivés dans les variantes respectives);

B_t et B_s, les ensembles d'opérateurs mis entre parenthèses dans les variantes b_t et b_s respectivement;

$B_t \setminus B_s$, la différence des ensembles B_t et B_s;

$P(B_t \setminus B_s)$, la puissance de l'ensemble $B_t \setminus B_s$.

En effet, supposons qu'on a déjà choisi dans les blocs $Y_1, \ldots, Y_{j-1}, Y_{j+1}, \ldots, Y_m$, par un procédé quelconque, les variantes $b_1, \ldots, b_{j-1}, b_{j+1}, \ldots, b_m$ d'un recouvrement minimal; soit B l'ensemble des opérateurs dérivés dans ces variantes. Il reste à choisir, pour ce recouvrement, une variante dans le bloc Y_j. Si $B_t \subseteq B$, on a le cas le plus favorable pour b_t: en effet, b_t ajoute dans ce cas au recouvrement choisi au plus autant d'opérateurs dérivés (exactement $L_t - P(B_t)$) qu'à n'importe quel autre recouvrement. Alors la quantité $(L_t - L_s) - P(B_t \setminus B_s)$ est la plus petite différence entre les nombres d'opérateurs dérivés apportés par les variantes b_t et b_s à un recouvrement quelconque du tableau des variantes, à condition que celles dans les autres blocs du tableau aient été choisies. Si la différence indiquée est non négative, la variante b_s est autant bonne (sinon meilleure) que b_t ($b_s \leqslant b_t$), si bien qu'on peut éliminer la variante b_t du tableau des variantes sans risque de perte de recouvrement minimal.

Il est évident que la relation $b_s \leqslant b_t$ est une relation d'ordre partiel. Nous dirons aussi que b_s et b_t ne sont pas ordonnés si $b_s \not\leqslant b_t$ et $b_t \not\leqslant b_s$, et équivalents si $b_s \leqslant b_t$ et $b_t \leqslant b_s$.

A titre d'exemple, comparons quelques variantes qui proviennent du tableau 7-6.

a) b_2 et b_3: $L_2 = 6$, $L_3 = 7$, $L_2 < L_3$;

$$(L_3 - L_2) - P(B_3 \setminus B_2) = (7-6) - P(\{14, 11, 10\} \setminus \{11, 10\}) =$$
$$= 1 - 1 = 0;$$
$$b_2 \leqslant b_3.$$

*) En toute rigueur, la première condition est une conséquence de la seconde; de ce fait, la condition 2) est suffisante.

b) b_2 et b_8: $L_2 = 6$, $L_8 = 6$, $L_2 = L_8$;

$$(L_8 - L_2) - P(B_8 \setminus B_2) = (6-6) - P(\{27, 14, 10\} \setminus \{11, 10\}) = \\ = 0-2 = -2;$$

$$b_2 \nleqslant b_8.$$

Vérifions la réciproque:

$$(L_2 - L_8) - P(B_2 \setminus B_8) = (6-6) - P(\{11, 10\} \setminus \{27, 14, 10\}) = \\ = 0 - 1 = -1;$$

$$b_8 \nleqslant b_2.$$

Les variantes b_2 et b_8 ne sont pas comparables.

c) b_4 et b_9: $L_4 = 4$, $L_9 = 4$, $L_4 = L_9$;

$$(L_9 - L_4) - P(B_9 \setminus B_4) = (4-4) - P(\{30, 12\} \setminus \varnothing) = 0 - 2 = \\ = -2;$$

$$b_4 \nleqslant b_9.$$

Vérifions la réciproque:

$$(L_4 - L_9) - P(B_4 \setminus B_9) = (4-4) - P(\varnothing \setminus \{30, 12\}) = 0 - 0 = \\ = 0;$$

$$b_9 \leqslant b_4.$$

d) En comparant b_{21} et b_{26}, on voit que $b_{21} \leqslant b_{26}$ et $b_{26} \leqslant b_{21}$; ce sont donc des variantes équivalentes.

L'algorithme de simplification du tableau des variantes est immédiat. On compare entre elles les variantes de réalisation de chaque bloc Y_j ($j = 1, \ldots, m$) et on en retient seulement celles qui ne sont pas moins bonnes que les autres. S'il y a deux variantes équivalentes parmi celles qui restent, on peut s'en limiter à une seule. Après une telle réduction, il ne reste dans chaque bloc Y_j ($j = 1, \ldots, m$) que des variantes non comparables. La réduction du tableau 7-6 conduit au tableau 7-7 dans lequel Y_1 et Y_3 contiennent une variante chacun, tandis que Y_2 contient deux variantes: b_{14} et b_{15}. Or, si les opérateurs 27, 14 et 10 figuraient entre parenthèses dans la variante b_{14} du tableau 7-6, ils apparaissent dans le tableau 7-7 sans parenthèses, car ils ne figurent en aucune variante pour Y_1 et Y_3. En comparant b_{14} et b_{15} dans le tableau 7-7, on constate que $b_{15} \leqslant b_{14}$: une nouvelle réduction conduit au tableau 7-8 qui contiendra une seule variante de sous-graphe de SGA, à savoir celle qui possède le plus petit nombre de sommets conditionnels. Le sous-graphe correspondant à neuf sommets conditionnels est tracé sur la figure 7-7.

Tableau 7-7

Première condensation de la table des variantes

Y_1	Y_2		Y_3
b_1	b_{14}	b_{15}	b_{19}
29	63	64	68
(12)	17	17	46
13	27	30	24
	14	(12)	(12)
	10		

Tableau 7-8

Recouvrement minimal de la table des variantes

Y_1	Y_2	Y_3
b_1	b_{15}	b_{19}
29	64	68
(12)	17	46
13	30	24
	(12)	(12)

Ainsi donc, la procédure de simplification doit être faite à plusieurs reprises. Si, après toutes les réductions possibles, il reste plus d'une variante dans un ou plusieurs blocs Y_j ($j = 1, \ldots, m$), on peut faire appel à la méthode de Petrick modifiée ou bien à une

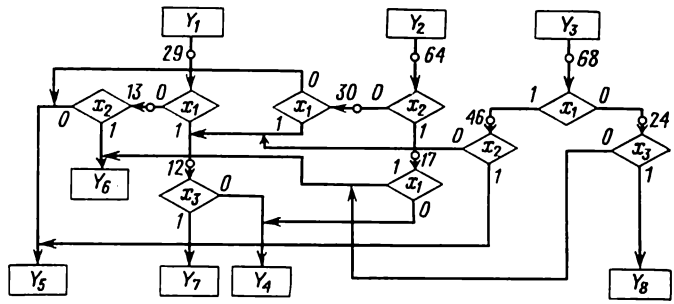


Fig. 7-7. Sous-graphe minimal équivalent au sous-graphe de la figure 7-3

autre méthode quelconque applicable à la recherche d'un recouvrement minimal du tableau de Quine, vu que, après simplification, le nombre de termes à manipuler se trouve considérablement réduit.

Pour le cas où le SGA serait incomplètement spécifié, nous introduirons la notion de recouvrement marqué incomplètement spécifié; il sera constitué de cubes du type c^sZ, où Z est le symbole d'indétermination et c^s le cube sur lequel le recouvrement en question est incomplètement spécifié. L'algorithme de recherche des sous-graphes équivalents reste le même, à ceci près que l'opération $c^rY_l \times c^sZ$ donne un cube c^pY_l, où c^p se définit comme précédemment, mais la coordonnée x suivant laquelle s'opère le recollement n'est pas soulignée (de même que dans le cas $c^rY_l \times c^sY_l$). En supposant que, dans le SGA de la figure 7-3, tous les recouvrements D_1, D_2 et D_3 sont incomplètement spécifiés sur le cube $10x \mid Z$, la procédure de

minimisation nous conduit à un sous-graphe à sept sommets conditionnels.

Il est à noter que si le SGA est incomplètement spécifié, l'algorithme ne garantit pas l'obtention d'une solution minimale. Par exemple, en minimisant le sous-graphe défini par les recouvrements

$$D_0^1 = \left| \begin{array}{cc|c} 0 & 0 & 3 \\ 1 & 1 & 5 \\ 1 & 0 & Z \\ 0 & 1 & Z \end{array} \right|, \quad D_0^2 = \left| \begin{array}{cc|c} 0 & 0 & 3 \\ 0 & 0 & 4 \\ x & 1 & 5 \end{array} \right|, \tag{7-6}$$

nous obtenons le sous-graphe montré sur la figure 7-8, *a*. Le sous-graphe minimal correspondant à ce cas est donné sur la figure 7-8, *b*.

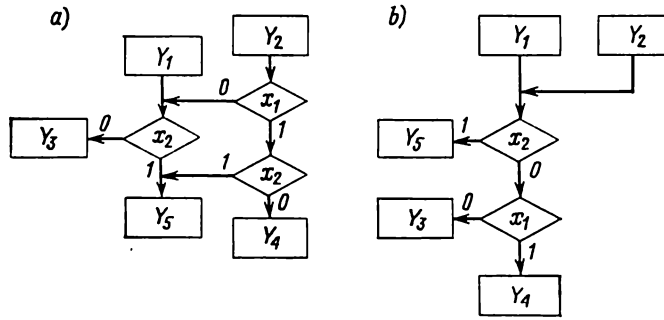


Fig. 7-8. Sous-graphes réalisant les recouvrements (7-6) : non minimal (*a*), minimal (*b*)

Dans des cas peu compliqués (comprenant jusqu'à cinq ou six conditions logiques), on peut se servir d'un tableau analogue à la table de Karnaugh dont les cases contiennent les opérateurs qui marquent les 0-cubes correspondants. Pour les recouvrements D_0^1, D_0^2 et D_0^3 (voir formules (7-5)) les tables de Karnaugh se présentent comme suit :

x_1 \ x_2x_3	00	01	11	10
0	5	5	6	6
1	4	7	7	4

D_0^1

x_1 \ x_2x_3	00	01	11	10
0	5	5	4	4
1	4	7	6	6

D_0^2

x_1 \ x_2x_3	00	01	11	10
0	6	8	8	6
1	4	7	5	5

D_0^3

et pour les recouvrements D_0^1 et D_0^2 (voir formules (7-6)), comme suit :

x_1 \ x_2	0	1
0	3	Z
1	Z	5

D_0^1

x_1 \ x_2	0	1
0	3	5
1	4	5

D_0^2

Les deux derniers tableaux permettent de voir que pour obtenir le sous-graphe minimal de la figure 7-8, *b*, on doit lever l'indétermination sur D_0^1 à l'aide de D_0^2, en remplaçant dans D_0^1 les cubes indéterminés 10 | Z et 01 | Z par 10 | 4 et 01 | 5 respectivement.

En minimisant un sous-graphe qui contient un sommet conditionnel à boucle (fig. 7-9), on introduit dans G_0 un opérateur fictif Y_w

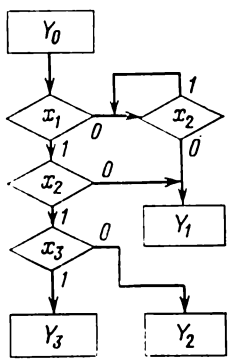


Fig. 7-9. Sous-graphe avec sommet à boucle avant la transformation

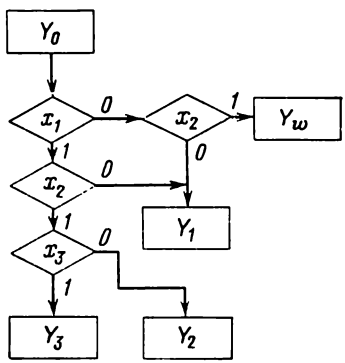


Fig. 7-10. Sous-graphe muni d'un opérateur fictif Y_w

auquel on joint la sortie rebouclée du sommet en question (fig. 7-10). A part cela, l'algorithme reste inchangé ; cependant, s'il y a un opérateur dérivé $Y_q = x_i Y_m Y_w$ ($Y_q = x_i Y_w Y_m$), la sortie 0 (1) du sommet conditionnel en x_i doit être reliée à l'entrée de celui-ci. Pour

le sous-graphe de la figure 7-10 on a

$$D_0^0 = \left|\begin{array}{ccc|c} 0 & 0 & x & 1 \\ 1 & 0 & x & 1 \\ 1 & 1 & 0 & 2 \\ 1 & 1 & 1 & 3 \\ 0 & 1 & x & w \end{array}\right|, \quad D_1^0 = \left|\begin{array}{ccc|c} 1 & 1 & \underline{x} & 4 \\ \underline{x} & 1 & 1 & 5 \\ 1 & \underline{x} & 1 & 6 \\ \underline{x} & 1 & 0 & 7 \\ 1 & \underline{x} & 0 & 8 \\ 0 & \underline{x} & x & 9 \\ x & 0 & x & 1 \\ 0 & 1 & x & w \end{array}\right|, \quad D_2^0 = \left|\begin{array}{ccc|c} 1 & \underline{x} & \underline{x} & 10, 13 \\ \underline{x} & 1 & \underline{x} & 11, 17 \\ \underline{x} & \underline{x} & 1 & 12, 14 \\ \underline{x} & \underline{x} & 0 & 15, 16 \\ 0 & \underline{x} & x & 9 \\ x & 0 & x & 1 \end{array}\right|,$$

$$D_3^0 = |\underline{x}\underline{x}\underline{x}|18,\ 19,\ 20,\ 21,\ 22,\ 23,\ 24,\ 25.$$

Les opérateurs dérivés sont résumés dans le tableau 7-9. Choisissant la variante minimale dans le tableau 7-10, on obtient le

Tableau 7-9

Opérateurs dérivés

4	x_3	3	2
5	x_1	3	w
6	x_2	3	1
7	x_1	2	w
8	x_2	2	1
9	x_2	w	1
10	x_2	4	1
11	x_3	5	7

12	x_2	5	1
13	x_3	6	8
14	x_1	6	9
15	x_2	7	1
16	x_1	8	9
17	x_1	4	w
18	x_1	13	9
19	x_2	11	1

20	x_3	12	15
21	x_3	12	16
22	x_3	14	15
23	x_3	14	16
24	x_1	10	9
25	x_2	17	1

Tableau 7-10

Table des variantes

b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
18	19	20	21	22	23	24	25
13	11	12	12	14	14	10	17
9	5	15	16	15	16	9	4
6	7	5	5	6	6	4	
8		7	8	9	9		
			9	7	8		

sous-graphe minimal de la figure 7-11. Il est intéressant de noter que les sommets à boucle des figures 7-9 et 7-11 contiennent des conditions logiques différentes.

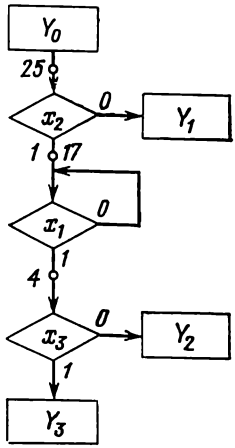


Fig. 7-11. Sous-graphe minimal avec un sommet à boucle

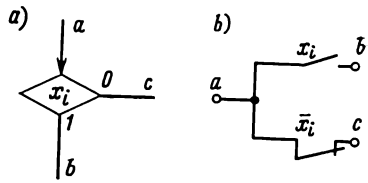


Fig. 7-12. Analogie entre un sommet conditionnel (*a*) et un contact de commutation (*b*)

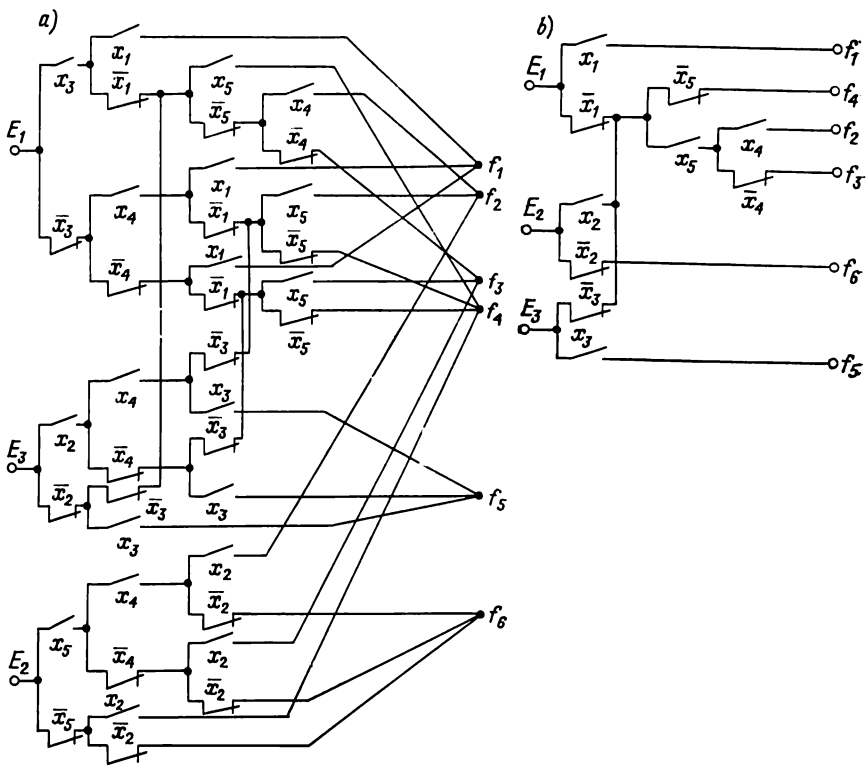


Fig. 7-13. Circuit à contacts de commutation: avant la minimisation (*a*), après la minimisation (*b*)

Bien qu'adaptés à la minimisation des schémas-graphes, les algorithmes décrits ci-dessus peuvent être utilisés tels quels, à la minimisation des circuits à relais ayant une structure d'arbre : ce sont des circuits à contacts de commutation à sorties multiples dont les fonctions de sortie forment des ensembles de valeurs disjoints. La figure 7-12 montre que d'un point de vue fonctionnel, un sommet conditionnel et un relais de commutation sont identiques. Elle permet de voir que les méthodes de minimisation des fonctions booléennes dans la classe des formes normales disjonctives ne conviennent pas à la minimisation des arbres à relais. La figure 7-13 donne un exemple de circuit à relais à six sorties avant et après la minimisation. Les pôles d'entrée E_1, E_2, E_3 jouent le rôle d'opérateurs de départ de sous-graphe ; les fonctions $f_1, \ldots, f_6$ constituent l'ensemble G_0 d'opérateurs primaires.

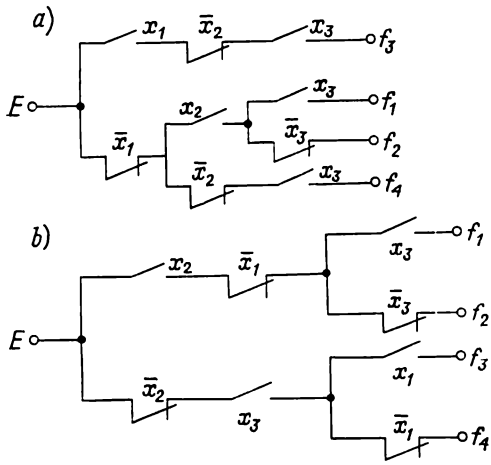


Fig. 7-14. Arbre à relais partiel : avant la minimisation (*a*), après la minimisation (*b*)

La minimisation des arbres à relais partiels est analogue à celle des sous-graphes présentant des sommets conditionnels à boucle. On voit sur la figure 7-14, *a* un arbre à relais partiel non minimal (cet exemple est tiré de [45]). Le recouvrement marqué initial est

$$\left|\begin{array}{ccc} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \\ x & 0 & 0 \\ 1 & 1 & x \\ 1 & x & 0 \end{array}\right| \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ w \\ w \\ w \end{array}$$

En appliquant l'algorithme exposé ci-dessus, on aboutit au circuit minimal de la figure 7-14, *b*.

7-2. Utilisation de la distribution des décalages pour la minimisation des schémas opératoriels d'algorithmes

En analysant le travail concret d'un système séquentiel dont le fonctionnement est défini par le schéma-graphe d'algorithme, il est bien souvent possible de relever les conditions logiques qui changent (ou restent inchangées) pendant l'exécution de tel ou tel opérateur. Par exemple, si un ordinateur exécute à chaque instant au plus une instruction, c'est-à-dire s'il n'y a pas de parallélisme local [40], le code de l'instruction inscrit dans le registre des instructions y reste inchangé jusqu'à l'arrivée d'une nouvelle instruction.

Associons à chaque opérateur Y_t ($t = 0, 1, \ldots, T + 1$; Y_0 et Y_{T+1} sont l'opérateur initial et l'opérateur final) d'un SGA ou SMA un sous-ensemble B_t de conditions logiques dont les éléments peuvent changer pendant l'exécution de l'opérateur Y_t: on a alors ce qu'on appelle une *distribution des décalages*. C'est une information complémentaire sur la logique du système qui vient compléter celle fournie par le SGA ou le SMA ; on peut la mettre à profit pour simplifier le SGA (SMA).

Si à chaque opérateur Y_t est associé l'ensemble complet des conditions logiques ($B_t = X = \{x_1, \ldots, x_L\}$), on dit que la distribution des décalages est *universelle*. Au contraire, si l'on a $B_t = \varnothing$ pour tout $t = 0, 1, \ldots, T + 1$ (aucune condition logique ne change pendant l'exécution des opérateurs), on dit que la distribution des décalages est *vide*.

Tableau 7-11

SMA M avant une minimisation par distribution des décalages

M	Y_1	Y_2	Y_3	Y_4	Y_5	Y_{fin}
Y_0	$\bar{x}_1$		$x_1\bar{x}_2$		x_1x_2	
Y_1	$\bar{x}_1x_2\bar{x}_3$		x_1x_2		$\bar{x}_1x_2x_3$	$\bar{x}_2$
Y_2				$\bar{x}_2x_4$	x_2x_4	$\bar{x}_4$
Y_3	$\bar{x}_1\bar{x}_4$			$\bar{x}_1x_4$		x_1
Y_4					x_2x_4	$\bar{x}_2 \vee \bar{x}_4$
Y_5		1				

Soit donnée pour le SMA M (tableau 7-11) une distribution des décalages (l'exemple est tiré de [15]):

$$\begin{aligned} Y_0 \to B_0 &= \{x_1, x_2, x_3, x_4\}, \\ Y_1 \to B_1 &= \{x_2, x_3, x_4\}, \\ Y_2 \to B_2 &= \varnothing, \\ Y_3 \to B_3 &= \{x_3\}, \\ Y_4 \to B_4 &= \{x_1, x_2, x_3\}, \\ Y_5 \to B_5 &= \{x_1, x_4\}, \\ Y_{\text{fin}} \to B_{\text{fin}} &= \{x_1, x_2, x_3, x_4\}. \end{aligned}$$

En regardant bien la colonne Y_1, nous remarquons que l'opérateur Y_1 ne devient exécutable qu'après les opérateurs Y_0, Y_1 ou Y_3, mais toujours avec $x_1 = 0$, car la variable x_1 n'intervient dans la colonne Y_1 que complémentée. Examinons la distribution des décalages pour Y_1: nous voyons que $x_1 \notin B_1$, c'est-à-dire que x_1 reste inchangée pendant l'exécution de Y_1, donc reste nulle après l'exécution de Y_1. On peut donc mettre 0 à la place de x_1 dans la ligne Y_1. Examinons sous ce point de vue toutes les autres colonnes et faisons les changements dans les lignes Y_3 ($x_1 = 1$), Y_4 ($x_4 = 1$) et Y_5 ($x_2 = 1$): nous obtenons alors le SMA M' (tableau 7-12) minimisé suivant le critère de la distribution des décalages.

Tableau 7-12

SMA M' après la minimisation du SMA M

M'	Y_1	Y_2	Y_3	Y_4	Y_5	Y_{fin}
Y_0	$\bar{x}_1$		$x_1\bar{x}_2$		x_1x_2	
Y_1	$x_2\bar{x}_3$				x_2x_3	$\bar{x}_2$
Y_2				$\bar{x}_2x_4$	x_2x_4	$\bar{x}_4$
Y_3						1
Y_4					x_2	$\bar{x}_2$
Y_5		1				

La procédure décrite se prête facilement à une formalisation. On trouve dans [15] un algorithme facile qui permet d'utiliser la distribution des décalages pour la minimisation des SMA. Il consiste à chercher, pour chaque colonne Y_t ($t = 1, \ldots, T + 1$) du schéma matriciel, une fonction f_{Y_t} égale à la disjonction des fonctions de transition qui figurent dans la colonne donnée:

$$f_{Y_t} = \bigvee_{j=0}^{T} \alpha_{jt}.$$

Dans le SMA M du tableau 7-11 on a

$$\begin{aligned} f_{Y_1} &= \bar{x}_1, & f_{Y_4} &= x_4(\bar{x}_1 \vee \bar{x}_2), \\ f_{Y_2} &= 1, & f_{Y_5} &= x_2(x_1 \vee x_3 \vee x_4), \\ f_{Y_3} &= x_1, & f_{Y_{\text{fin}}} &= x_1 \vee \bar{x}_2 \vee \bar{x}_4. \end{aligned}$$

Parmi les f_{Y_t} ($t = 1, \ldots, T + 1$), on cherche des *fonctions particulières réduites* par rapport aux variables $x_{i1}, \ldots, x_{iR} \in X$, c'est-à-dire des fonctions qui se laissent mettre sous la forme $x_{i1}^{e_{i1}} \ldots x_{iR}^{e_{iR}} f' = \varphi_{Y_t} f'$, où f' ne dépend pas de $x_{i1}, \ldots, x_{iR}$ ($e_{ir} \in \{0, 1\}$; $x_{ir}^1 = x_{ir}$, $x_{ir}^0 = \bar{x}_{ir}$; $r = 1, \ldots, R$). Eliminant de φ_{Y_t} les conditions logiques qui interviennent dans la distribution des décalages de l'opérateur Y_t, on obtient les fonctions $\varphi^*_{Y_t}$. De toute évidence, si $\varphi^*_{Y_t} = x_{h1}^{e_{h1}} \ldots x_{hQ}^{e_{hQ}}$ ($\{x_{h1}, \ldots, x_{hQ}\} \subseteq \{x_{i1}, \ldots, x_{iR}\}$), on peut, dans les expressions α_{tp} ($p = 1, \ldots, T + 1$) de la ligne Y_t, donner à la variable x_{hq} la valeur 1 si $e_{hq} = 1$, et la valeur 0 si $e_{hq} = 0$ ($q = 1, \ldots, Q$).

Utilisant la distribution des décalages, on tire directement du tableau 7-11 les fonctions suivantes : $\varphi^*_{Y_1} = \bar{x}_1$, $\varphi^*_{Y_3} = x_1$, $\varphi^*_{Y_4} = x_4$, $\varphi^*_{Y_5} = x_2$. En donnant la valeur 0 à x_1 dans la ligne Y_1, la valeur 1 à x_1 dans la ligne Y_3, la valeur 1 à x_4 dans la ligne Y_4 et la valeur 1 à x_2 dans la ligne Y_5, on obtient le SMA M' tel qu'il figure dans le tableau 7-12.

Dans le SMA M', la fonction $f_{Y_2} = 1$ n'est pas une fonction particulière réduite; elle ne peut donc pas être utilisée telle pour la minimisation du SMA. Or, le tableau 7-12 permet de voir que Y_2 devient exécutable après Y_5, quelles que soient les conditions logiques. On a d'autre part $\varphi^*_{Y_5} = x_2$, c'est-à-dire que la variable x_2 prend la valeur 1 avant l'exécution de Y_5 et conserve cette valeur immédiatement après. Portons donc $\alpha_{52} = x_2$ au lieu de $\alpha_{52} = 1$ dans la ligne Y_5: il vient $\varphi^*_{Y_2} = x_2$. Donnons ensuite la valeur 1 à

Tableau 7-13

SMA M'' après une nouvelle minimisation du SMA M'

M''	Y_1	Y_2	Y_3	Y_5	Y_{fin}
Y_0	$\bar{x}_1$		$x_1\bar{x}_2$	x_1x_2	
Y_1	$x_2\bar{x}_3$			x_2x_3	$\bar{x}_2$
Y_2				x_4	$\bar{x}_4$
Y_3					1
Y_5		1			

x_2 dans la ligne Y_2: il vient $\alpha_{52} = 1$ comme précédemment. Le SMA se présente sous sa forme définitive M'' donnée par le tableau 7-13. On remarque que ce dernier ne comporte pas la colonne Y_4 et la ligne Y_4, car tous les α_{j4} ($j = 0, 1, \ldots, 5$) sont nuls.

7-3. Minimisation du schéma-graphe d'algorithme par réduction des sommets opératoriels

La minimisation d'un SGA complètement spécifié peut être effectuée par une méthode analogue à celle qui est proposée dans [21] pour la minimisation des microprogrammes *). Soit un SGA (fig. 7-15) muni d'un sommet initial, d'un sommet final, de dix sommets opératoriels et de six sommets conditionnels. Nous allons considérer que ce schéma-graphe traduit un automate de Moore dont les états sont identifiés aux sommets initial, final et opératoriels du graphe et dont la fonction des transitions est définie de la façon proposée dans le § 8-2 pour la synthèse d'un automate de Moore microprogrammé d'après son SGA. Comme signal de sortie, nous prendrons l'opérateur inscrit dans le sommet qui correspond à l'état considéré, en précisant que le passage à l'état initial a_1 est accompagné par le signal de sortie Y_0, et le passage à l'état final a_{fin}, par le signal Y_{fin}.

*) Pour bien assimiler le matériel du présent paragraphe, la lecture du § 8-2 est indispensable.

La table directe des transitions de l'automate de Moore microprogrammé est donnée dans le tableau 7-14. Puisque le SGA minimisé doit posséder un sommet initial et un sommet final, nous ferons du sommet final du SGA de départ un état indépendant. Puisque aucun chemin ne part du sommet final du SGA, on ne peut passer de l'état a_{fin} de l'automate de Moore à aucun autre état ; autrement dit, une

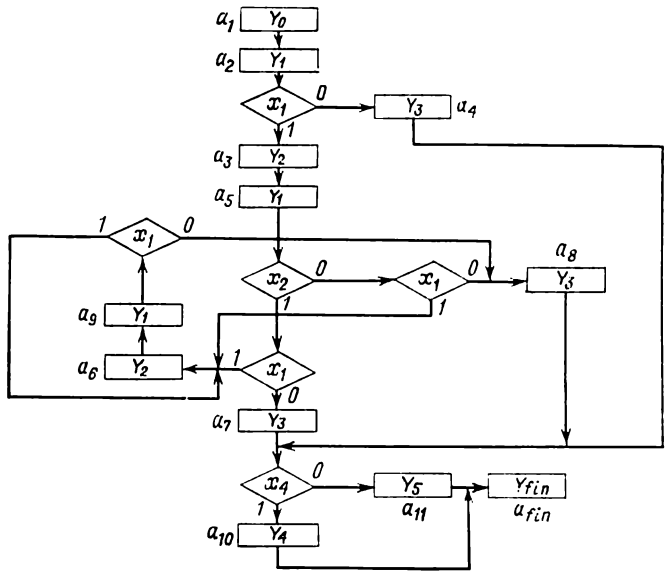


Fig. 7-15. Un SGA avant la minimisation suivant le nombre de sommets opératoriels

fois mis dans l'état a_{fin}, l'automate y reste quel que soit le signal d'entrée. Ce cas est résumé par la dernière ligne du tableau 7-14.

La minimisation de l'automate consiste à faire des partitions successives de l'ensemble de ses états internes en classes d'états 0-, 1-, . . ., p-, $(p+1)$-équivalents, jusqu'à ce que deux partitions consécutives soient identiques. Pour l'automate décrit dans le tableau 7-14 on a

$$\pi_0 = \overline{1}, \overline{2\cdot5\cdot9}, \overline{3\cdot6}, \overline{4\cdot7\cdot8}, \overline{10}, \overline{11}, \overline{\text{fin}} = \{A_0, A_1, A_2, A_3, A_4, A_5, A_6\}$$

(voir tableau 7-15).

Du tableau 7-15 on déduit

$$\pi_1 = \overline{1}, \overline{2\cdot5\cdot9}, \overline{3\cdot6}, \overline{4\cdot7\cdot8}, \overline{10}, \overline{11}, \overline{\text{fin}}; \ \pi_0 = \pi_1.$$

Tableau 7-14

Automate de Moore correspondant au SGA de la figure 7-15

Etat initial (et signal de sortie)	Etat de transition	Signal d'entrée
$a_1\,(Y_0)$	a_2	1
$a_2\,(Y_1)$	a_3 a_4	x_1 $\bar{x}_1$
$a_3\,(Y_2)$	a_5	1
$a_4\,(Y_3)$	a_{10} a_{11}	x_4 $\bar{x}_4$
$a_5\,(Y_1)$	a_6 a_6 a_7 a_8	x_2x_1 $\bar{x}_2x_1$ $x_2\bar{x}_1$ $\bar{x}_2\bar{x}_1$
$a_6\,(Y_2)$	a_9	1
$a_7\,(Y_3)$	a_{10} a_{11}	x_4 $\bar{x}_4$
$a_8\,(Y_3)$	a_{10} a_{11}	x_4 $\bar{x}_4$
$a_9\,(Y_1)$	a_6 a_8	x_1 $\bar{x}_1$
$a_{10}\,(Y_4)$	a_{fin}	1
$a_{11}\,(Y_5)$	a_{fin}	1
$a_{fin}\,(Y_{fin})$	a_{fin}	1

Tableau 7-15

Partition π_0

Classe d'équivalence	Etat initial (et signal de sortie)	Classe de transition	Signal d'entrée
A_0	$a_1\,(Y_0)$	A_1	1
A_1	$a_2\,(Y_1)$	A_2	x_1
		A_3	$\bar{x}_1$
	$a_5\,(Y_1)$	A_2	x_2x_1
		A_3	$x_2\bar{x}_1$
		A_2	$\bar{x}_2x_1$
		A_3	$\bar{x}_2\bar{x}_1$
	$a_9\,(Y_1)$	A_2	x_1
		A_3	$\bar{x}_1$
A_2	$a_3\,(Y_2)$	A_1	1
	$a_6\,(Y_2)$	A_1	1
A_3	$a_4\,(Y_3)$	A_4	x_4
		A_5	$\bar{x}_4$
	$a_7\,(Y_3)$	A_4	x_4
		A_5	$\bar{x}_4$
	$a_8\,(Y_3)$	A_4	x_4
		A_5	$\bar{x}_4$
A_4	$a_{10}\,(Y_4)$	A_6	1
A_5	$a_{11}\,(Y_5)$	A_6	1
A_6	$a_{\text{fin}}\,(Y_{\text{fin}})$	A_6	1

La suite des partitions est terminée. Choisissant un état dans chaque classe de π_1, on obtient la table des transitions de l'automate de Moore minimal défini dans le tableau 7-16. Il s'agit maintenant

Tableau 7-16

Automate de Moore minimal

Etat initial (et signal de sortie)	Etat de transition	Signal d'entrée
$a_1\,(Y_0)$	a_2	1
$a_2\,(Y_1)$	a_3 a_4	x_1 $\bar{x}_1$
$a_3\,(Y_2)$	a_2	1
$a_4\,(Y_3)$	a_{10} a_{11}	x_4 $\bar{x}_4$
$a_{10}\,(Y_4)$	a_{fin}	1
$a_{11}\,(Y_5)$	a_{fin}	1
$a_{fin}\,(Y_{fin})$	a_{fin}	1

de passer de cette table à un schéma-graphe équivalent au schéma-graphe initial mais possédant un sommet initial, un sommet final et cinq sommets opératoriels. A cet effet, en procédant comme dans le § 7-1, on partitionne le SGA futur en sous-graphes (fig. 7-16) *).

Conformément au nombre de sous-graphes, on obtient quatre sous-systèmes de formules de transition :

$$\left.\begin{aligned} &Y_0 \to Y_1,\ Y_2 \to Y_1\,; \\ &Y_1 \to x_1 Y_2 \vee \bar{x}_1 Y_3\,; \\ &Y_3 \to x_4 Y_4 \vee \bar{x}_4 Y_5\,; \\ &Y_4 \to Y_{fin},\ Y_5 \to Y_{fin}. \end{aligned}\right\} \qquad (7\text{-}7)$$

*) On pourrait passer du tableau 7-16 de façon triviale à un SMA, puis du SMA au SGA ; on peut aussi le faire directement à l'aide du tableau 7-16.

Ensuite, si nous voulons minimiser aussi les sommets conditionnels des sous-graphes obtenus, nous devons appliquer la méthode de minimisation décrite dans le § 7-1 ou bien chercher (par examen complet) un système de formules de transition parenthétisées proche

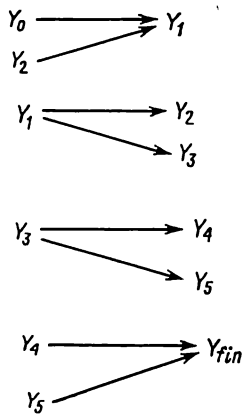


Fig. 7-16. Décomposition conduisant au SGA ayant le plus petit nombre de sommets opératoriels

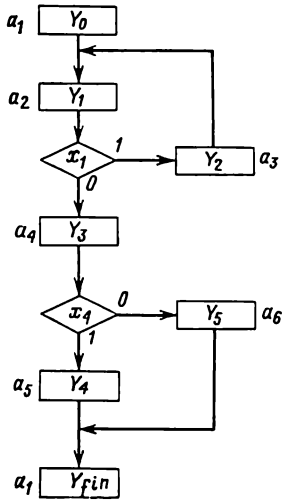


Fig. 7-17. SGA minimisé suivant le nombre de sommets opératoriels

d'un système minimal. Le système (7-7) est déjà un système de formules parenthétisées; c'est aussi un système minimal, on le voit en examinant les expressions dégagées. Le schéma-graphe d'algorithme tracé d'après le système (7-7) est donné sur la figure 7-17.

7-4. Fusionnement des schémas-graphes d'algorithmes

En donnant la description de l'algorithme de fonctionnement d'un système complexe, il y a souvent intérêt à construire plusieurs SGA partiels dont chacun définit une partie du comportement général du système. Par exemple, quand il s'agit de l'unité de commande d'un ordinateur, la solution la plus facile consiste à tracer un schéma-graphe pour chaque opération. Ces schémas-graphes auront évidemment certains sommets opératoriels et conditionnels en commun; si l'on réalise un automate de commande séparé pour chaque SGA, on a très peu de chances d'aboutir à un résultat optimal. On a donc à résoudre le problème du fusionnement des SGA partiels en un schéma-graphe unique, ce qui permettra en fin de

compte de minimiser le nombre de sommets opératoriels et conditionnels.

Soient des SGA $\Gamma_1, \ldots, \Gamma_Q$ qui peuvent présenter certains opérateurs communs, sans qu'un opérateur se répète à l'intérieur d'un même SGA. A chaque SGA Γ_q $(q = 1, \ldots, Q)$ nous ferons correspondre sa propre collection $(e_{q1}, \ldots, e_{qN})$ de valeurs des variables $p_1, \ldots, p_N$ (chaque collection étant différente des autres), où $e_{qn} \in \{0, 1\}$, $N \geqslant]\log_2 Q[$, $p_n \notin \bigcup_{q=1}^{Q} X_q$ (X_q est l'ensemble des conditions logiques du SGA Γ_q), $n = 1, \ldots, N$. Ici $]a[$ désigne comme précédemment le plus petit entier supérieur à a ou, si ce dernier est lui-même un entier, égal à a. Nous dirons que la conjonction $P_q = \bigwedge_{n=1}^{N} p_n^{e_{qn}}$ est *déterminante* [15] pour le SGA Γ_q. Le problème du fusionnement des SGA $\Gamma_1, \ldots, \Gamma_Q$ revient à construire un SGA Γ qui présente les deux propriétés suivantes:

1) tout opérateur Y_t figurant dans l'un au moins des SGA partiels Γ_q intervient obligatoirement, mais une fois tout au plus, dans le SGA Γ;

2) en portant dans le SGA Γ une collection de valeurs $(e_{q1}, \ldots, e_{qN})$ sur laquelle $P_q = 1$, le SGA Γ se transforme en un SGA équivalent au SGA partiel Γ_q (voir [38]).

L'exposé de l'algorithme sera illustré sur l'exemple de fusionnement des trois sous-graphes Γ_1, Γ_2, Γ_3 montrés sur la figure 7-18. La construction du graphe fusionné se fait par étapes:

1. Pour chaque SGA Γ_q construire un SMA correspondant M_q. Dans notre exemple ce seront les SMA M_1, M_2 et M_3 (voir tableaux 7-17, 7-18 et 7-19).

2. Coder chaque SMA M_q avec un vecteur $(e_{q1}, \ldots, e_{qN})$, $e_{qn} \in \{0, 1\}$. La question du codage des SMA partiels sera discutée en fin du paragraphe, après avoir étudié l'algorithme de fusionnement, de manière à mieux cerner l'influence des codes choisis sur la complexité du SGA.

Nous coderons les SMA M_1, M_2 et M_3 comme suit:

$$K(M_1) = (00),\ K(M_2) = (11),\ K(M_3) = (01).$$

3. A chaque SMA M_q faire correspondre une conjonction $P_q = p_1^{e_{q1}} \ldots p_N^{e_{qN}}$, où $(e_{q1}, \ldots, e_{qN})$ est le code du SMA M_q; $p_n^0 = \bar{p}_n$, $p_n^1 = p_n$. Dans notre exemple

$$P_1 = \bar{p}_1\bar{p}_2,\ P_2 = p_1p_2,\ P_3 = \bar{p}_1p_2.$$

4. Construire le SMA fusionné M dont les lignes et les colonnes correspondent à tous les opérateurs qui figurent dans l'union des ensembles d'opérateurs des SMA $M_1, \ldots, M_Q$. Comme précé-

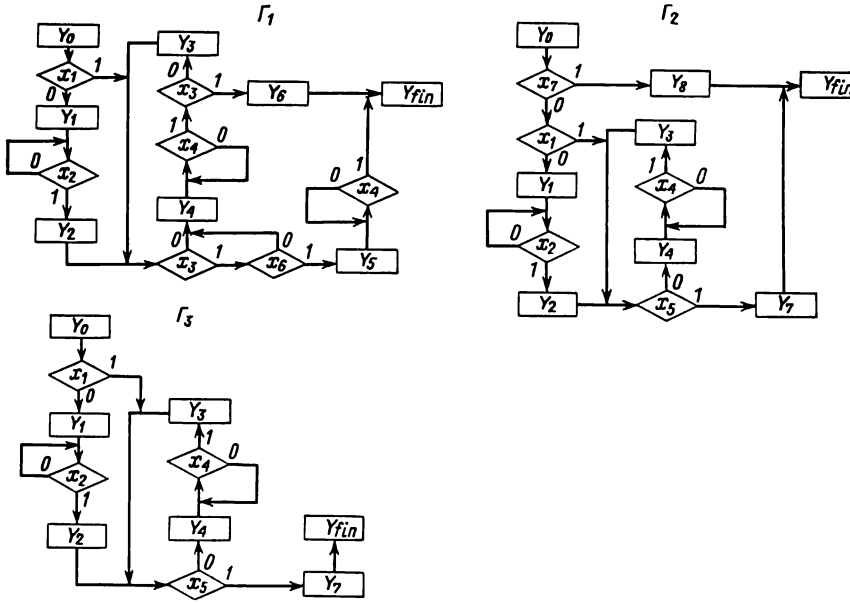


Fig. 7-18. Trois schémas-graphes Γ_1, Γ_2 et Γ_3 à fusionner

Tableau 7-17

SMA M_1

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_{fin}
Y_0	$\bar{x}_1$			$x_1\bar{x}_3$ $x_1x_3\bar{x}_6$	$x_1x_3x_6$		
Y_1		x_2					
Y_2				$\bar{x}_3$ $x_3\bar{x}_6$	x_3x_6		
Y_3				$\bar{x}_3$ $x_3\bar{x}_6$	x_3x_6		
Y_4			$x_4\bar{x}_3$			x_4x_3	
Y_5							x_4
Y_6							1

Tableau 7-18

SMA M_2

	Y_1	Y_2	Y_3	Y_4	Y_7	Y_8	Y_{fin}
Y_0	$\bar{x}_7\bar{x}_1$			$\bar{x}_7x_1\bar{x}_5$	$\bar{x}_7x_1x_5$	x_7	
Y_1		x_2					
Y_2				$\bar{x}_5$	x_5		
Y_3				$\bar{x}_5$	x_5		
Y_4			x_4				
Y_7							1
Y_8							1

Tableau 7-19

SMA M_3

	Y_1	Y_2	Y_3	Y_4	Y_7	Y_{fin}
Y_0	$\bar{x}_1$			$x_1\bar{x}_5$	x_1x_5	
Y_1		x_2				
Y_2				$\bar{x}_5$	x_5	
Y_3				$\bar{x}_5$	x_5	
Y_4			x_4			
Y_7						1

demment, aucune ligne n'y est réservée à l'opérateur final Y_{fin} et aucune colonne à l'opérateur initial Y_0. Chaque élément α_{ij} du SMA M est égal à $\bigvee_{q=1}^{Q} \alpha_{ij}^q P_q$, où α_{ij}^q est l'élément du SMA M_q qui se trouve à l'intersection de la ligne Y_i et de la colonne Y_j.

Le SMA fusionné M construit à partir des SMA M_1, M_2 et M_3 est donné dans le tableau 7-20.

Puisque la conjonction P_q ($q = 1, \ldots, Q$) est égale à 1 toutes les fois que le SMA M se comporte comme le SMA M_q, aucun opérateur ne change les valeurs des variables $p_1, \ldots, p_N$ (la distribution des décalages par rapport à ces variables est vide), si bien qu'on a la possibilité de simplifier le SMA M. Par exemple, la transition vers l'opérateur Y_5 (colonne Y_5 du tableau 7-20) s'opère toujours avec $p_1 = p_2 = 0$ ($\varphi^*_{Y_5} = \bar{p}_1\bar{p}_2$), ceci nous permet de remplacer, dans la ligne Y_5, les variables p_1 et p_2 par $p_1 = p_2 = 0$. D'une façon analogue, on peut mettre $p_1 = p_2 = 0$ dans la ligne Y_6, $p_2 = 1$ dans la ligne Y_7, $p_1 = p_2 = 1$ dans la ligne Y_8: on obtient un SMA M' minimisé à l'aide de la distribution des décalages (tableau 7-21).

Pour passer du SMA M' au SGA fusionné Γ, on doit partager le SMA M' en sous-matrices et appliquer la méthode de construction du SGA minimal proposée dans le § 7-1, ou bien chercher (par examen complet) un système de formules de transition parenthétisées proche d'un système minimal pour chacune des sous-matrices. Dans le premier cas comme dans le second, on doit tenir compte des indéterminations de deux types qui apparaissent quand on réunit les SGA partiels:

1. Si le nombre Q des SGA à fusionner est inférieur à 2^N, c'est-à-dire s'il y a des collections de valeurs des variables $p_1, \ldots, p_N$ qui n'ont pas été utilisées pour le codage des SMA $M_1, \ldots, M_Q$, toutes les formules de transition (ou les recouvrements marqués des formules de transition) sont non définies sur les collections de valeurs des variables $p_1, \ldots, p_N, x_1, \ldots, x_L$ qui admettent comme recouvrement les cubes $e_{s1} \ldots e_{sN}x \ldots x$, où $e_{s1} \ldots e_{sN}$ est la collection non utilisée de valeurs des variables $p_1, \ldots, p_N$. Dans l'exemple considéré on n'utilise pas la collection p_1, p_2 égale à 10: cela revient à dire que toutes les formules de transition sont non définies sur le cube $10x \ldots x$.

2. Si un opérateur Y_t n'intervient pas dans un SMA M_q, la formule de transition pour Y_t (recouvrement marqué de Y_t) n'est pas définie sur les collections de valeurs des variables qui admettent comme recouvrement le cube $e_{q1} \ldots e_{qN}x \ldots x$, où $e_{q1} \ldots e_{qN}$ est le code du SMA M_q. Dans notre exemple Y_5 est absent dans M_2 et M_3, aussi la formule de transition pour Y_5 est-elle non définie sur les cubes $11x \ldots x$ et $01x \ldots x$. Tout pareillement, la fonction

Tableau 7-20

SMA fusionné *M*

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_{fin}
Y_0	$\bar{p}_1\bar{p}_2\bar{x}_1$ $p_1p_2\bar{x}_7\bar{x}_1$ $\bar{p}_1p_2\bar{x}_1$			$\bar{p}_1\bar{p}_2x_1\bar{x}_3$ $\bar{p}_1\bar{p}_2x_1x_3\bar{x}_6$ $p_1p_2\bar{x}_7x_1\bar{x}_5$ $\bar{p}_1p_2x_1\bar{x}_5$	$\bar{p}_1\bar{p}_2x_1x_3x_6$		$p_1p_2\bar{x}_7x_1x_5$ $\bar{p}_1p_2x_1x_5$	$p_1p_2x_7$	
Y_1		$\bar{p}_1\bar{p}_2x_2$ $p_1p_2x_2$ $\bar{p}_1p_2x_2$							
Y_2				$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1\bar{p}_2x_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{x}_5$	$\bar{p}_1\bar{p}_2x_3x_6$		$p_1p_2x_5$ $\bar{p}_1p_2x_5$		
Y_3				$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1\bar{p}_2x_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{x}_5$	$\bar{p}_1\bar{p}_2x_3x_6$		$p_1p_2x_5$ $\bar{p}_1p_2x_5$		
Y_4			$\bar{p}_1\bar{p}_2x_4\bar{x}_3$ $p_1p_2x_4$ $\bar{p}_1p_2x_4$			$\bar{p}_1\bar{p}_2p_1x_3$			
Y_5									$\bar{p}_1\bar{p}_2x_4$
Y_6									$\bar{p}_1\bar{p}_2$
Y_7									p_1p_2 $\bar{p}_1p_2$
Y_8									p_1p_2

Tableau 7-21

SMA M' minimisé par distribution des décalages

	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_{fin}
Y_0	$\bar{p}_1\bar{p}_2\bar{x}_1$ $p_1p_2\bar{x}_7\bar{x}_1$ $\bar{p}_1p_2\bar{x}_1$			$\bar{p}_1\bar{p}_2x_1\bar{x}_3$ $\bar{p}_1\bar{p}_2x_1x_3\bar{x}_6$ $p_1p_2\bar{x}_7x_1\bar{x}_5$ $\bar{p}_1p_2x_1\bar{x}_5$	$\bar{p}_1\bar{p}_2x_1x_3x_6$		$p_1p_2\bar{x}_7x_1x_5$ $\bar{p}_1p_2x_1x_5$	$p_1p_2x_7$	
Y_1		$\bar{p}_1\bar{p}_2x_2$ $p_1p_2x_2$ $\bar{p}_1p_2x_2$							
Y_2				$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1\bar{p}_2x_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{p}_5$	$\bar{p}_1\bar{p}_2x_3x_6$		$p_1p_2x_5$ $\bar{p}_1p_2x_5$		
Y_3				$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1\bar{p}_2x_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{x}_5$	$\bar{p}_1\bar{p}_2x_3x_6$		$p_1p_2x_5$ $\bar{p}_1p_2x_5$		
Y_4			$\bar{p}_1\bar{p}_2x_4\bar{x}_3$ $p_1p_2x_4$ $\bar{p}_1p_2x_4$			$\bar{p}_1\bar{p}_2x_4x_3$			
Y_5									x_4
Y_6									1
Y_7									p_1 $\bar{p}_1$
Y_8									1

Tableau 7-22

Sous-matrice M'_1

	Y_1	Y_4	Y_5	Y_7	Y_8
Y_0	$\bar{p}_1\bar{p}_2\bar{x}_1$ $p_1p_2\bar{x}_7\bar{x}_1$ $\bar{p}_1p_2\bar{x}_1$	$\bar{p}_1\bar{p}_2x_1\bar{x}_3$ $\bar{p}_1\bar{p}_2x_1x_3\bar{x}_6$ $p_1p_2\bar{x}_7x_1\bar{x}_5$ $\bar{p}_1p_2x_1\bar{x}_5$	$\bar{p}_1\bar{p}_2x_1x_3x_6$	$p_1p_2\bar{x}_7x_1x_5$ $\bar{p}_1p_2x_1x_5$	$p_1p_2x_7$
Y_2		$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1\bar{p}_2x_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{x}_5$	$\bar{p}_1\bar{p}_2x_3x_6$	$p_1p_2x_5$ $\bar{p}_1p_2x_5$	
Y_3		$\bar{p}_1\bar{p}_2\bar{x}_3$ $\bar{p}_1p_2\bar{x}_3\bar{x}_6$ $p_1p_2\bar{x}_5$ $\bar{p}_1p_2\bar{x}_5$	$\bar{p}_1\bar{p}_2x_3x_6$	$p_1p_2x_5$ $\bar{p}_1p_2x_5$	

Tableau 7-23

Sous-matrice M'_2

	Y_2
Y_1	$\bar{p}_1\bar{p}_2x_2$ $p_1p_2x_2$ $\bar{p}_1p_2x_2$

Tableau 7-24

Sous-matrice M'_3

	Y_3	Y_6
Y_4	$\bar{p}_1\bar{p}_2x_4\bar{x}_3$ $p_1p_2x_4$ $\bar{p}_1p_2x_4$	$\bar{p}_1\bar{p}_2x_4x_3$

de transition est non définie pour Y_6 sur les cubes $11x \ldots x$ et $01x \ldots x$, pour Y_7 sur $00x \ldots x$ et pour Y_8 sur $00x \ldots x$ et $01x \ldots x$.

Les sous-matrices obtenues par partition du SGA M' sont données dans les tableaux 7-22, 7-23, 7-24 et 7-25.

En construisant pour les sous-matrices M'_1, M'_2, M'_3 et M'_4 les sous-graphes correspondants Γ^1, Γ^2, Γ^3 et Γ^4 et en opérant leur fusionnement suivant les sommets opératoriels prédécesseurs et successeurs identiques, on obtient le SGA fusionné Γ montré sur la figure 7-19. Ce SGA est construit par les méthodes décrites dans le § 7-1. On pourrait obtenir le même résultat en explicitant les formules de transition à partir des sous-matrices, en les spécifiant sur les collections non utilisées de valeurs des variables, puis en minimisant les formules de transition et en écrivant le système de formules parenthétisées. Il n'existe malheureusement aucun algorithme qui puisse garantir l'obtention d'une solution proche de la solution minimale à la suite de ces transformations des formules de transition. Les difficultés les plus considérables apparaissent quand les formules de transition sont incomplètement spécifiées : pour obtenir une bonne solution dans ces conditions, il faut une certaine expérience et des essais réitérés. Par exemple, pour aboutir au schémagraphe Γ de la figure 7-19, on doit définir complètement les formules de transition à l'aide des expressions mises entre crochets, puis les transformer en un système de formules parenthétisées de la façon suivante :

Tableau 7-25

Sous-matrice M'_4

	Y_{fin}
Y_5	x_4
Y_6	1
Y_7	p_1 $\bar{p}_1$
Y_8	1

$$
\begin{gathered}
Y_0 \to (\bar{p}_1\bar{p}_2\bar{x}_1 \vee p_1p_2\bar{x}_7\bar{x}_1 \vee \bar{p}_1p_2\bar{x}_1 \vee [p_1\bar{p}_2\bar{x}_7\bar{x}_1])\, Y_1 \vee \\
\vee (\bar{p}_1\bar{p}_2x_1\bar{x}_3 \vee \bar{p}_1\bar{p}_2x_1x_3\bar{x}_6 \vee p_1p_2\bar{x}_7x_1\bar{x}_5 \vee \bar{p}_1p_2x_1\bar{x}_5 \vee \\
\vee [p_1\bar{p}_2x_1\bar{x}_3\bar{x}_7 \vee p_1\bar{p}_2x_1x_3\bar{x}_6\bar{x}_7])\, Y_4 \vee (\bar{p}_1\bar{p}_2x_1x_3x_6 \vee \\
\vee [p_1\bar{p}_2x_1x_3x_6\bar{x}_7])\, Y_5 \vee (p_1p_2\bar{x}_7x_1x_5 \vee \bar{p}_1p_2x_1x_5)\, Y_7 \vee \\
\vee (p_1p_2x_7 \vee [p_1\bar{p}_2x_7])\, Y_8 = p_1\,(x_7Y_8 \vee \\
\vee \bar{x}_7\,(x_1\,(\underline{p_2}\,(x_5\underline{Y_7} \vee \underline{\bar{x}_5}Y_4 \vee \underline{\bar{p}_2}\,(x_3\,(\underline{x_6}Y_5 \vee \underline{\bar{x}_6}\underline{Y_4}) \vee \underline{\bar{x}_3}\underline{Y_4})) \vee \bar{x}_1Y_1)) \vee \\
\vee \bar{p}_1\,(x_1\,(\underline{p_2}\,(x_5\underline{Y_7} \vee \underline{\bar{x}_5}Y_4) \vee \underline{\bar{p}_2}\,(x_3\,(\underline{x_6}Y_5 \vee \underline{\bar{x}_6}\underline{Y_4}) \vee \underline{\bar{x}_3}\underline{Y_4})) \vee \bar{x}_1Y_1)\,; \\
Y_2 = Y_3 \to (\bar{p}_1\bar{p}_2\bar{x}_3 \vee \bar{p}_1\bar{p}_2x_3\bar{x}_6 \vee p_1p_2\bar{x}_5 \vee \bar{p}_1p_2\bar{x}_5 \vee \\
\vee [p_1\bar{p}_2x_3\bar{x}_6 \vee p_1\bar{p}_2\bar{x}_3])\, Y_4 \vee (\bar{p}_1\bar{p}_2x_3x_6 \vee \\
\vee [p_1\bar{p}_2x_3x_6])\, Y_5 \vee (p_1p_2x_5 \vee \bar{p}_1p_2x_5)\, Y_7 = \\
= \underline{p_2}\,(x_5\underline{Y_7} \vee \underline{\bar{x}_5}Y_4) \vee \underline{\bar{p}_2}\,(x_3\,(\underline{x_6}Y_5 \vee \underline{\bar{x}_6}\underline{Y_4}) \vee \underline{\bar{x}_3}\underline{Y_4})\,; \\
Y_1 \to (\bar{p}_1\bar{p}_2x_2 \vee p_1p_2x_2 \vee \bar{p}_1p_2x_2 \vee [p_1\bar{p}_2x_2])\, Y_2 = x_2Y_2\,;
\end{gathered}
$$

$$Y_4 \to (\bar{p}_1\bar{p}_2x_4\bar{x}_3 \vee p_1p_2x_4 \vee \bar{p}_1p_2x_4 \vee [p_1\bar{p}_2x_4\bar{x}_3])\, Y_3 \vee (\bar{p}_1\bar{p}_2x_4x_3 \vee$$
$$\vee [p_1\bar{p}_2x_4x_3])\, Y_6 = x_4\,(p_2Y_3 \vee \bar{p}_2\,(x_3Y_6 \vee \bar{x}_3Y_3))\,;$$

$$Y_5 \to x_4Y_{\text{fin}}\,;$$
$$Y_6 \to Y_{\text{fin}}\,;$$
$$Y_7 \to p_1Y_{\text{fin}} \vee \bar{p}_1Y_{\text{fin}} = Y_{\text{fin}}\,;$$
$$Y_8 \to Y_{\text{fin}}.$$

Il est évident que, dans ces expressions, aux mêmes formules partielles (on les a mis en relief parmi les expressions ci-dessus) correspondent les mêmes sous-graphes du schéma-graphe fusionné.

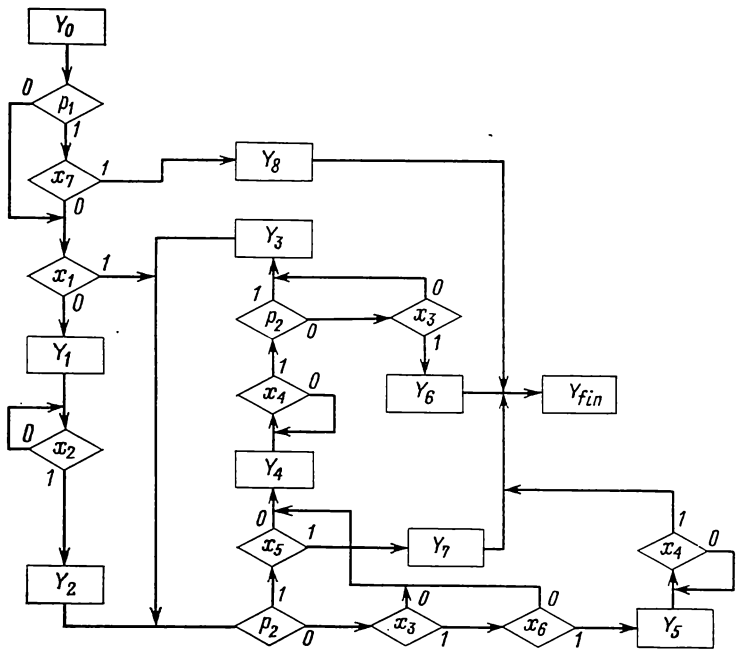


Fig. 7-19. Le SGA Γ obtenu par fusionnement des schémas-graphes Γ_1, Γ_2, Γ_3

En reprenant notre exemple, montrons que le problème de fusionnement des schémas-graphes se laisse diviser en plusieurs sous-problèmes de moindre dimension. A cet effet, faisons les partitions des SMA initiaux M_1, M_2, M_3 (fig. 7-20) en sous-matrices M_{11}, M_{12}, M_{13}, M_{14} (partition de M_1), M_{21}, M_{22}, M_{23}, M_{24} (partition M_2), M_{31}, M_{32}, M_{33}, M_{34} (partition de M_3). Sur l'ensemble de

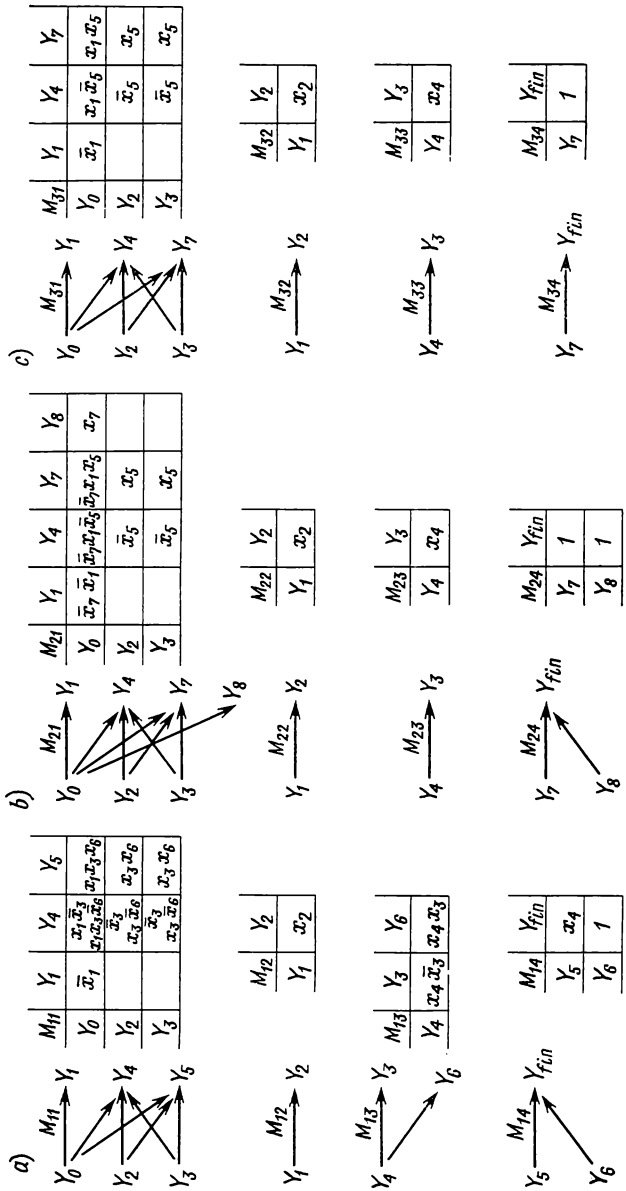
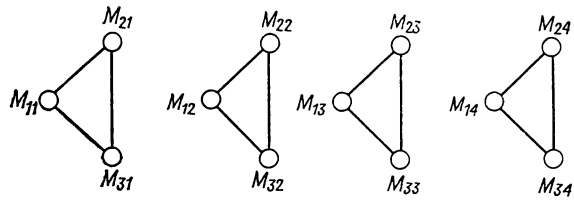


Fig. 7-20. Décomposition des SMA M_1 (*a*), M_2 (*b*), M_3 (*c*)

sous-matrices ainsi constitué, introduisons la relation de recollement en disant que deux sous-matrices M_t et M_s peuvent être *recollées*, $M_t \circ M_s$, si $((A_t \cap A_s) \neq \varnothing) \vee ((B_t \cap B_s) \neq \varnothing) = 1$, où A_t (B_t) et A_s (B_s) sont les ensembles des opérateurs prédécesseurs (successeurs) des sous-matrices M_t et M_s respectivement. Il est évident que cette relation est réflexive et symétrique mais n'est pas transitive. Donnons à cette relation la forme d'un graphe dont les sommets correspondront aux sous-matrices; toutes les fois que



Fir. 7-21. Graphe de la relation de recollement de sous-matrices

$M_t \circ M_s$, les sommets M_t et M_s seront reliés par une arête (fig. 7-21). Pour avoir l'ensemble de toutes les sous-matrices susceptibles d'être recollées, on doit évidemment chercher toutes les composantes connexes du graphe. Les matrices appartenant à une même composante sont à recoller. Le graphe de la figure 7-21 présente quatre composantes; on a donc à résoudre quatre problèmes partiels distincts, beaucoup plus faciles, qui consistent à chercher les possibilités de fusionnement des sous-matrices suivantes:

$$M_{11},\quad M_{21},\quad M_{31};$$
$$M_{12},\quad M_{22},\quad M_{32};$$
$$M_{13},\quad M_{23},\quad M_{33};$$
$$M_{14},\quad M_{24},\quad M_{34}.$$

Après avoir effectué le recollement et mis à profit la distribution des décalages, nous aboutissons directement aux sous-matrices M'_1, M'_2, M'_3, M'_4 (tableaux 7-22 à 7-25), sans passer par le SMA fusionné M'.

Revenons au problème du codage des SMA partiels $M_1, \ldots, M_Q$. De toute évidence, si l'on veut minimiser le nombre de sommets conditionnels marqués par des variables p_n $(n = 1, \ldots, N)$, on a intérêt à coder par des codes adjacents les SMA qui présentent le plus grand nombre d'éléments identiques. On doit en outre tenir compte de la complexité des éléments considérés. Pour pouvoir faire la comparaison, attribuons aux éléments de SMA α_{ij} des poids égaux au nombre des lettres de la fonction α_{ij}; ils seront notés b (α_{ij}). Par exemple, dans le SMA M_1 on a b $(\alpha_{01}) = 1$,

$b(\alpha_{04}) = 5$, $b(\alpha_{6\,\text{fin}}) = 1$. Quand $\alpha_{ij} = 1$, on admet qu'elle est de poids 1. Pour chaque couple de SMA M_q, M_r ($q, r = 1, \ldots, Q$; $q \neq r$), calculons la somme des poids des éléments identiques α_{ij} qui appartiennent à l'intersection de Y_i et Y_j dans les deux SMA M_q, M_r (ces sommes se notent B_{qr}). Dans notre exemple $B_{12} = 1$, $B_{13} = 2$, $B_{23} = 7$. Si un opérateur Y_t ne figure que dans M_q ou dans M_r, nous calculerons le poids total de tous les éléments non nuls de la ligne Y_t et nous ajouterons ce poids à B_{qr} : la somme obtenue, notée C_{qr}, sera appelée poids du couple (M_q, M_r). Dans notre exemple $C_{12} = 1 + 4 = 5$, $C_{13} = 2 + 3 = 5$, $C_{23} = 7 + 1 = 8$.

Il convient de coder par des codes adjacents en premier lieu les couples de SMA de poids maximal. Nous avons codé les SMA M_1, M_2 et M_3 comme suit : $K(M_1) = 00$, $K(M_2) = 11$, $K(M_3) = 01$.

Si l'on codait les SMA M_2 et M_3 par des codes non adjacents (en faisant par exemple $K(M_1) = 01$, $K(M_2) = 11$, $K(M_3) = 00$), le SGA fusionné aurait deux sommets conditionnels de plus. On s'en assure en examinant les formules de transition parenthétisées établies pour les opérateurs Y_2 et Y_4 :

$$Y_2 \to \bar{p}_1 p_2 \bar{x}_3 Y_4 \vee \bar{p}_1 p_2 x_3 \bar{x}_6 Y_4 \vee \bar{p}_1 p_2 x_3 x_6 Y_5 \vee p_1 p_2 \bar{x}_5 Y_4 \vee p_1 p_2 x_5 Y_7 \vee$$
$$\vee\, p_1 \bar{p}_2 \bar{x}_5 Y_4 \vee \bar{p}_1 \bar{p}_2 x_5 Y_7 \vee [p_1 \bar{p}_2 \bar{x}_5 Y_4 \vee p_1 \bar{p}_2 x_5 Y_7] =$$
$$= p_2 (\bar{p}_1 (\bar{x}_3 Y_4 \vee x_3 (\bar{x}_6 Y_4 \vee x_6 Y_5)) \vee p_1 \underline{(\bar{x}_5 Y_4 \vee x_5 Y_7)}) \vee \bar{p}_2 \underline{(\bar{x}_5 Y_4 \vee x_5 Y_7)}\,;$$
$$Y_4 \to \bar{p}_1 p_2 x_4 \bar{x}_3 Y_3 \vee \bar{p}_1 p_2 x_4 x_3 Y_6 \vee p_1 p_2 x_4 Y_3 \vee \bar{p}_1 \bar{p}_2 x_4 Y_3 \vee [p_1 \bar{p}_2 x_4 Y_3] =$$
$$= x_4 (\bar{p}_1 (p_2 (\bar{x}_3 Y_3 \vee x_3 Y_6) \vee \bar{p}_2 Y_3) \vee p_1 Y_3).$$

En établissant les formules de transition parenthétisées, il nous a fallu tenir compte de leur indétermination du fait que le code 10 n'a pas été utilisé pour le codage des schémas matriciels.

CHAPITRE 8

SYNTHÈSE DES AUTOMATES MICROPROGRAMMÉS D'APRÈS LE SCHÉMA-GRAPHE D'ALGORITHME

8-1. Synthèse d'un automate microprogrammé de Mealy

Par automate *microprogrammé* on entend un automate fini qui réalise le microprogramme de fonctionnement d'un système séquentiel [22, 38, 5]. La synthèse d'un automate microprogrammé d'après le schéma-graphe d'algorithme [5] s'effectue en deux temps:

1. Construction du SGA marqué.
2. Construction du graphe de l'automate.

La première étape consiste à attribuer des symboles a_1, a_2, . . ., aux entrées des sommets qui suivent les sommets opératoriels. L'attribution s'opère conformément aux règles ci-après:

a) le symbole a_1 est attribué à l'entrée du sommet qui suit le sommet initial, ainsi qu'à l'entrée du sommet final;

b) les entrées de tous les sommets qui suivent les sommets opératoriels doivent être marquées;

c) chaque entrée à marquer se voit attribuer un symbole et un seul;

d) les entrées des différents sommets (à l'exception du sommet final) sont à marquer par des symboles différents.

Il est clair que le marquage nécessite un nombre fini de symboles-marques. Supposons, pour fixer les idées, que les entrées des sommets du schéma-graphe ont été marquées par les symboles a_1, ..., a_M. La première étape de la synthèse se termine par l'établissement d'un SGA *marqué* qui sert de base pour la deuxième étape: le tracé du graphe de l'automate.

Appliquée au schéma-graphe d'algorithme de la figure 6-4, la première partie de cette procédure donne le SGA marqué que l'on voit sur la figure 8-1.

Allant d'une marque a_m à une autre a_s dans le sens d'orientation des arcs du SGA et inscrivant le contenu des sommets traversés, nous obtiendrons pour chaque chemin parcouru un mot dans l'alphabet

$$\{x_1^{e_1}, \ldots, x_L^{e_L}, Y_1, \ldots, Y_T\},$$

où

$$e_l = \begin{cases} 1 & \text{chaque fois qu'on quitte le sommet conditionnel par sa sortie 1,} \\ 0 & \text{chaque fois qu'on quitte le sommet conditionnel par sa sortie 0;} \end{cases}$$

$$x_l^0 = \bar{x}_l, \quad x_l^1 = x_l, \quad l = 1, \ldots, L.$$

Pour souligner que le mot obtenu correspond au chemin de a_m à a_s, nous encadrerons ce mot par a_m et a_s.

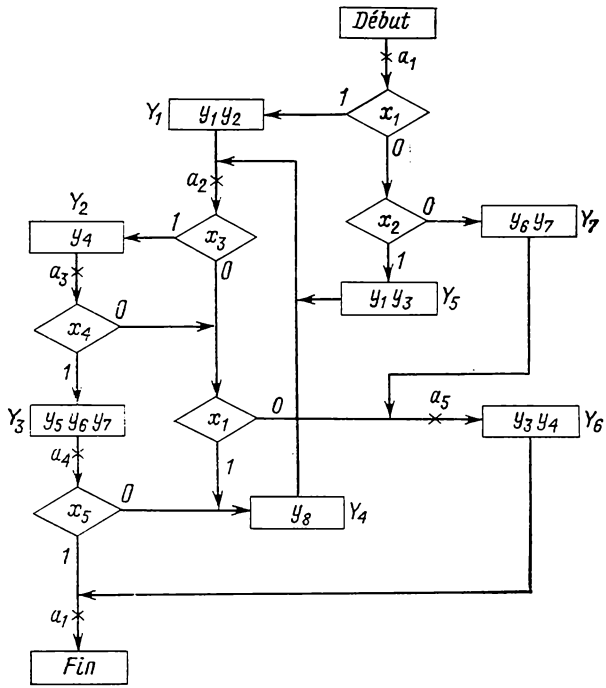


Fig. 8-1. SGA marqué utilisé pour la synthèse d'un automate de Mealy

Nous considérerons dans le texte qui suit des mots du type

$$a_m x_{m1}^{e_{m1}} \ldots x_{mR}^{e_{mR}} \; Y_t a_s \quad (a_m, a_s \in \{a_1, \ldots, a_M\}) \tag{8-1}$$

ou

$$a_m x_{m1}^{e_{m1}} \ldots x_{mR}^{e_{mR}} \; a_1 \quad (a_m \in \{a_2, a_3, \ldots, a_M\}). \tag{8-2}$$

Les chemins qui correspondent à ces mots sur le schéma-graphe seront appelés *chemins de transition*. Parmi les chemins du type (8-1),

on rencontre parfois le cas où $R = 0$:

$$a_m Y_t a_s. \tag{8-3}$$

Ainsi donc, un chemin du type (8-1) est un chemin sur le SGA qui va d'une marque a_m à une autre a_s (avec peut-être $a_m = a_s$) et qui contient exactement un sommet opératoriel. Un chemin du type (8-2) est un chemin qui va d'une marque a_m à la marque a_1 ($a_m \neq a_1$) et qui ne contient que des sommets conditionnels.

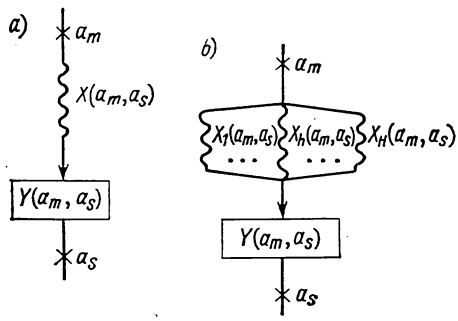


Fig. 8-2. Représentation conventionnelle des chemins de transition de a_m à a_s: un chemin unique (*a*), un ensemble de H chemins (*b*)

A chaque chemin (ou mot) du type (8-1) ou (8-2) on peut associer une conjonction

$$X(a_m, a_s) = \bigwedge_{r=1}^{R} x_{mr}^{e_{mr}}.$$

Nous désignerons brièvement ces chemins

$$a_m X(a_m, a_s) \times Y(a_m, a_s)\, a_s$$

$$\text{et } a_m X(a_m, a_1)\, a_1,$$

où $Y(a_m, a_s) = Y_t$.

On peut représenter schématiquement le chemin de transition de a_m à a_s de la façon indiquée sur la figure 8-2, *a*, où la ligne ondulée correspond au chemin traversant des sommets conditionnels.

Il y a des cas où l'on peut arriver de a_m en a_s par plusieurs chemins différents du type $a_m X_h(a_m, a_s)\, Y(a_m, a_s)\, a_s$ ($h = 1, \ldots, H$) qui passent tous par un même sommet opératoriel (voir fig. 8-2, *b*). Nous admettrons alors qu'à l'ensemble des chemins considéré correspond la disjonction $X(a_m, a_s) = \bigvee_{h=1}^{H} X_h(a_m, a_s)$; l'ensemble des chemins lui-même sera noté $a_m X(a_m, a_s)\, Y(a_m, a_s)\, a_s$ et appelé chemin de transition généralisé.

Ayant établi le SGA marqué, on passe à la construction du graphe de l'automate de Mealy S dont les états sont $a_1, \ldots, a_M$; cet automate est initialisé par a_1. Les transitions et les signaux de sortie de cet automate seront définis comme suit.

On cherche tous les chemins de transition sur le SGA marqué. Si, pour un certain $r = 1, \ldots, R$, le symbole $x_r^{e_r}$ intervient plus d'une fois dans le chemin, tous les symboles $x_r^{e_r}$ sauf un sont éliminés de ce chemin. Si pour un certain $r = 1, \ldots, R$ on rencontre dans un chemin tant x_r que $\bar{x}_r$, le chemin en question n'est plus considéré.

A chaque chemin de transition du type $a_m X\ (a_m,\ a_s)\ Y\ (a_m, a_s)\ a_s$, où $a_m,\ a_s \in \{a_1,\ \ldots,\ a_M\}$, on fait correspondre une transition de l'automate S de l'état a_s induite par le signal d'entrée $X\ (a_m,\ a_s)$ et accompagnée par la production du signal de sortie $Y\ (a_m,\ a_s)$. A chaque chemin de transition du type $a_m Y\ (a_m,\ a_s)\ a_s$, où a_m, $a_s \in \{a_1,\ \ldots,\ a_M\}$, on fait correspondre une transition de l'automate S de l'état a_m à l'état a_s, où a_m, a_s sont définis comme précédemment. Le signal d'entrée pour cette transition est égal comme précédemment à la conjonction des contenus des sommets conditionnels traversés. Puisque dans le cas considéré l'ensemble des sommets conditionnels est vide et que la conjonction d'un ensemble de variables vide est égale à 1, nous admettons que le signal d'entrée est égal à 1, tandis que le signal de sortie est $Y\ (a_m,\ a_s)$ comme dans le cas précédent.

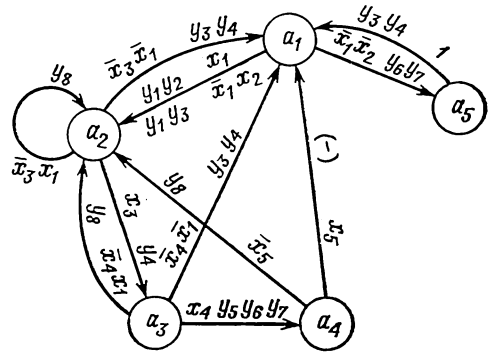


Fig. 8-3. Graphe de l'automate de Mealy correspondant au SGA de la figure 8-1

A chaque chemin de transition du type $a_m X\ (a_m,\ a_1)\ a_1$ on fait correspondre la transition de a_m à a_1; dans ce cas l'état a_m et le signal d'entrée sont définis comme précédemment, tandis que le signal de sortie est égal à Y_0 (opérateur vide).

On obtient en résultat le graphe de l'automate de Mealy S qui admet autant d'états qu'il y a de symboles attribués aux entrées des sommets du SGA. Le graphe de l'automate de Mealy correspondant au SGA de la figure 8-1 est tracé sur la figure 8-3.

Si l'automate S possède L voies d'entrée, ses signaux d'entrée sont des vecteurs à L composantes $\Delta_1,\ \ldots,\ \Delta_{2^L}$, où $\Delta_j = (e_{j1},\ \ldots\ \ldots,\ e_{jL})$ et la composante e_{jl} $(j = 1,\ \ldots,\ 2^L;\ l = 1,\ \ldots,\ L)$ peut prendre les valeurs 0 ou 1. D'autre part, s'il existe sur le SGA entre a_m et a_s un chemin de transition $a_m X\ (a_m,\ a_s)\ Y\ (a_m,\ a_s)\ a_s$ qui traverse le sommet opératoriel $Y\ (a_m,\ a_s)$, on associe à l'arc du graphe de l'automate le signal d'entrée $X\ (a_m,\ a_s)$ et le signal de sortie $Y\ (a_m,\ a_s)$.

Les automates microprogrammés réalisés par le procédé décrit présentent tous un trait particulier, à savoir : chaque signal d'entrée associé à un arc du graphe est un produit élémentaire, c'est-à-dire une fonction booléenne des variables figurant dans le chemin de transition considéré ; il est à noter que la longueur de ce produit est sensiblement moins grande que le nombre total L des variables d'en-

trée. Il est évident que la transition (a_m, a_s) accompagnée par l'élaboration du signal de sortie $Y(a_m, a_s)$ sera induite par les collections d'entrée de l'ensemble $\{\Delta_1, \ldots, \Delta_{2^L}\}$ sur lesquelles la fonction booléenne $X(a_m, a_s)$ prend la valeur 1.

Dans un automate déterministe, le produit de deux signaux d'entrée quelconques induisant une transition à partir d'un état de l'automate doit être égal à 0. Dans un automate construit d'après un SGA, cette condition est toujours remplie. En effet, si un état a_m admet plus d'une transition vers d'autres états, on peut, en choisissant de façon arbitraire deux transitions possibles, par exemple vers les états a_s et a_t (avec peut-être $s = t$, ce qui veut dire que les deux transitions choisies conduisent à un même état), s'assurer que l'on rencontre dans les conjonctions $X(a_m, a_s)$ et $X(a_m, a_t)$ une variable quelconque, par exemple x_r, à la fois sous forme directe (x_r) et complémentée $(\bar{x}_r)$. Cela tient à ce que deux chemins de transition $a_m X(a_m, a_s) Y(a_m, a_s) a_s$ et $a_m X(a_m, a_t) Y(a_m, a_t) a_t$ admettent toujours un sommet conditionnel en commun contenant x_r; dans le premier chemin on quitte ce sommet par la sortie 1, et dans le second, par la sortie 0. On peut donc associer à chaque état a_m de l'automate S une partition de l'ensemble des signaux d'entrée (c'est-à-dire des collections $\Delta_1, \ldots, \Delta_{2^L}$ de valeurs des variables dans les voies d'entrée de l'automate) en $R + 1$ classes disjointes deux à deux $E_{m1}, \ldots, E_{mr}, \ldots, E_{m,R+1}$, où R est le nombre des conjonctions différentes $X_{m1}, \ldots, X_{mr}, \ldots, X_{mR}$ qui sont associées aux arcs du graphe de S incidents à a_m vers l'extérieur. Si l'on n'a qu'un seul arc issu de a_m et marqué par le signal 1 (le cas du chemin $a_m Y(a_m, a_s) a_s$ par exemple, lorsque le sommet $Y(a_m, a_s)$ suit immédiatement un autre sommet opératoriel), on admet que $R = 0$.

Il est évident qu'à toute collection $\Delta_j = (e_{j1}, \ldots, e_{jL})$ correspond de façon univoque un constituant de l'unité $x_1^{e_{j1}} \ldots x_L^{e_{jL}}$, c'est-à-dire une fonction booléenne de L variables qui prend la valeur 1 sur la collection Δ_j et la valeur 0 sur toutes les autres collections. Ceci étant, nous ferons correspondre à chacune des R premières classes E_{mr} $(r = 1, \ldots, R)$ seulement les collections de valeurs des variables d'entrée dont les constituants de l'unité correspondants sont absorbés par la conjonction X_r qui définit la classe en question, c'est-à-dire que

$$E_{mr} = \{\Delta_j \mid X_{mr} \vee x_1^{e_{j1}} \ldots x_L^{e_{jL}} = X_{mr}\}, \tag{8-4}$$

où

$$r = 1, \ldots, R;\ j \in \{1, \ldots, 2^L\};\ e_{jl} \in \{0, 1\};$$
$$x_l^0 = \bar{x}_l,\ x_l^1 = x_l;\ l = 1, \ldots, L.$$

Toutes les autres collections seront rapportées à la $(R+1)$-ième classe :

$$E_{m,\,R+1} = \{\Delta_1, \ldots, \Delta_{2L}\} \setminus \bigcup_{r=1}^{R} E_{mr}. \tag{8-5}$$

Si $R = 0$, on forme une classe unique qui regroupe toutes les 2^L collections.

Une partition en classes ainsi définie de l'ensemble des collections $\Delta_1, \ldots, \Delta_{2L}$ permet de distinguer les transitions induites par chacune des collections. Puisque la fonction X_{mr} admet la valeur 1 sur l'ensemble des collections E_{mr} $(r = 1, \ldots, R)$, la transition de l'automate de l'état a_m sous l'effet de n'importe quel signal d'entrée, contenu dans l'une des E_{mr} classes, et le signal de sortie correspondant seront respectivement les mêmes que la transition induite par la conjonction associée à la classe donnée et le signal de sortie qui se produit lors de cette transition. Au contraire, attaqué par n'importe quelle collection appartenant à la $E_{m,\,R+1}$-ième classe, l'automate reste dans le même état a_m et produit le signal de sortie $Y_0 = (0 \ldots 0)$. Si $R = 0$, l'automate attaqué par n'importe quelle collection d'entrée passe au même état et produit le même signal de sortie que sous l'action du signal 1 associé à l'arc du graphe de l'automate.

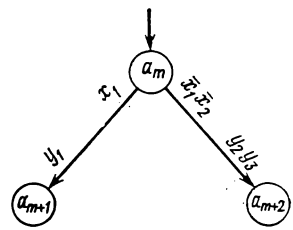


Fig. 8-4. Sous-graphe d'un automate

Illustrons les raisonnements développés ci-dessus sur un exemple. Soit un automate à trois voies d'entrée $(L = 3)$ et à quatre voies de sortie $(N = 4)$. Supposons qu'un état a_m de cet automate admet deux transitions vers d'autres états (fig. 8-4). Le nombre R des produits élémentaires sur les arcs issus de a_m est égal à 2 $(X_{m1} = x_1,\ X_{m2} = \bar{x}_1\bar{x}_2)$. Il existe donc huit collections d'entrée possibles ; on les partitionne en $R + 1 = 3$ classes E_{m1}, E_{m2}, E_{m3}, dont les deux premières se laissent définir par (8-4) et la dernière par (8-5) :

$$E_{m1} = \{111,\ 110,\ 101,\ 100\},\ E_{m2} = \{001,\ 000\}\,;\ E_{m3} = \{010,\ 011\}.$$

Introduisons la notion de *réalisation du schéma-graphe* de l'automate de commande. Soit un automate structuré S possédant L voies d'entrée et N voies de sortie. Mettons-le à l'état initial a_1 et appliquons à ses entrées une séquence de collections à L composantes $\Delta_{m1}, \ldots, \Delta_{mQ}$ du type $(e_{q1}, \ldots, e_{qL})$, $q = 1, \ldots, Q$, $e_{ql} \in \{0, 1\}$, $l = 1, \ldots, L$. Supposons que l'automate attaqué par cette séquence d'entrée produise une séquence de sortie de collections à N composantes $\Omega_{i1}, \ldots, \Omega_{iQ}$ du type $(e_{q1}, \ldots, e_{qN})$,

$q = 1, \ldots, Q$, $e_{qn} \in \{0, 1\}$, $n = 1, \ldots, N$. Faisons une séquence d'ensembles $Y_{i1}, \ldots, Y_{iQ}$ tels que $y_n \in Y_{iq}$ si dans la collection Ω_{iq} on a $e_{qn} = 1$ ($q = 1, \ldots, Q$; $n = 1, \ldots, N$). Cette séquence sera appelée réponse de l'automate structuré S à la séquence d'entrée $\Delta_{m1}, \ldots, \Delta_{mQ}$. Nous dirons que l'automate S *réalise* un schéma-graphe d'algorithme Γ si ses réponses à toutes les séquences possibles de collections avant le retour à l'état a_1 se confondent avec les valeurs que prend le SGA sur les mêmes séquences, aux opérateurs Y_0 et Y_{fin} près (ici Y_0 et Y_{fin} sont les opérateurs qui encadrent chaque valeur du SGA).

Cherchons la réponse que présente l'automate S (voir fig. 8-3) se trouvant dans l'état a_1 à la séquence de collections

Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6
10101	11011	10110	01101	10100	01011

pour laquelle on a déterminé dans le § 6-2 la valeur $Y_0Y_1Y_4Y_2Y_6Y_{\text{fin}}$ du SGA :

Collections d'entrée	Δ_1	Δ_2	Δ_3	Δ_4	Δ_5	Δ_6
Séquence d'états	a_1	a_2	a_3	a_3	a_1	a_2
Réponse	Y_1	Y_4	Y_2	Y_6	Y_1	

Nous voyons que la réponse de l'automate, jusqu'à son retour à l'état a_1, se confond avec la valeur que prend le SGA (sans Y_0 et Y_{fin}) sur la même séquence de collections.

Du procédé décrit de construction de l'automate microprogrammé de Mealy d'après le schéma-graphe d'algorithme Γ, il ressort immédiatement que l'automate microprogrammé ainsi construit réalise effectivement le SGA Γ.

Dans le texte qui suit, nous considérerons (à moins d'indication contraire) les automates synchronisés par des impulsions d'horloge. Autrement dit, l'automate possède, en plus des voies d'entrée $x_1, \ldots, x_L$, au moins une voie d'horloge p ; on a $p = 1$ quand l'impulsion d'horloge est présente et $p = 0$ quand elle est absente. De ce fait, nous retiendrons comme signal d'entrée pour la transition (a_m, a_s) correspondant au chemin $a_mX\ (a_m, a_s)\ Y\ (a_m, a_s)\ a_s$, non pas $X\ (a_m, a_s)$ mais la conjonction $pX\ (a_m, a_s)$. Nous conviendrons cependant de n'associer le signal p à aucun arc du graphe de l'automate.

8-2. Synthèse d'un automate microprogrammé de Moore

La synthèse d'un automate de Moore d'après le schéma-graphe d'algorithme s'opère aussi en deux temps : l'obtention du SGA marqué et le tracé du graphe de l'automate [5]. La première phase

de la synthèse consiste à attribuer des symboles $a_1, \ldots, a_M$ aux sommets initial, final et opératoriels d'après les règles suivantes:

1. Associer le symbole a_1 aux sommets initial et final.
2. Marquer les sommets opératoriels différents par des symboles différents.
3. Chaque sommet opératoriel doit être marqué.

A la différence de l'automate de Mealy, en construisant un automate de Moore, on ne marque pas les entrées des sommets qui sui-

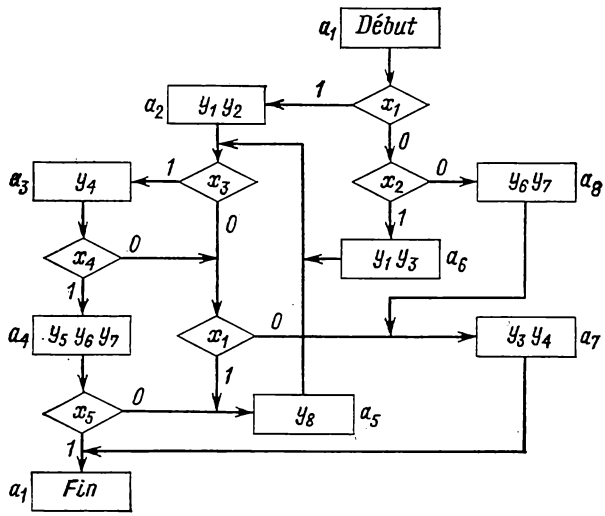


Fig. 8-5. SGA marqué utilisé pour la synthèse de l'automate de Moore

vent les sommets opératoriels mais les sommets opératoriels eux-mêmes. Le nombre des marques s'avère donc d'une unité supérieur à celui des sommets opératoriels sur le schéma-graphe d'algorithme. A la suite de la procédure décrite, on obtient, à partir du SGA de la figure 6-4, le schéma-graphe marqué représenté sur la figure 8-5.

On commence le tracé du graphe de l'automate de Moore en cherchant sur le SGA marqué des chemins de transition du type

$$a_m x_{m1}^{e_{m1}} \ldots x_{mR}^{e_{mR}} a_s, \tag{8-6}$$

où e_{mr} $(r = 1, \ldots, R)$ et $x_{mr}^{e_{mr}}$ sont déterminés comme dans l'expression (8-1). Comme précédemment, nous adopterons pour ces chemins la notation abrégée $a_m X (a_m, a_s) a_s$; s'il existe entre a_m et a_s un ensemble de chemins du type $a_m X_h (a_m, a_s) a_s$ $(h = 1, \ldots, H)$, nous ferons correspondre à cet ensemble de chemins la disjonction $X (a_m, a_s) = \bigvee_{h=1}^{H} X_h (a_m, a_s)$, nous noterons cet ensem-

ble de chemins $a_m X\ (a_m,\ a_s)\ a_s$ et nous l'appellerons chemin de transition généralisé.

Avec des chemins du type (8-6), on a parfois $R = 0$, quand deux sommets opératoriels sont séparés par un ensemble vide de sommets conditionnels (le sommet a_s suit immédiatement le sommet a_m, si bien que le chemin du type (8-6) se réduit au chemin $a_m a_s$).

Tableau 8-1

Table des transitions directe de l'automate de Mealy

a_m	a_s	$X\ (a_m,\ a_s)$	$Y\ (a_m,\ a_s)$
a_1	a_2	x_1	$y_1 y_2$
	a_2	$\bar{x}_1 x_2$	$y_1 y_3$
	a_5	$\bar{x}_1 \bar{x}_2$	$y_6 y_7$
a_2	a_1	$\bar{x}_3 \bar{x}_1$	$y_3 y_4$
	a_2	$\bar{x}_3 x_1$	y_8
	a_3	x_3	y_4
a_3	a_1	$\bar{x}_4 \bar{x}_1$	$y_3 y_4$
	a_2	$\bar{x}_4 x_1$	y_8
	a_4	x_4	$y_5 y_6 y_7$
a_4	a_1	x_5	—
	a_2	$\bar{x}_5$	y_8
a_5	a_1	1	$y_3 y_4$

Tableau 8-2

Table des transitions inverse de l'automate de Moore

a_m	$a_s,\ Y\ (a_s)$	$X\ (a_m,\ a_s)$
a_4	a_1 (—)	x_5
a_7		1
a_1	$a_2\ (y_1 y_2)$	x_1
a_2	$a_3\ (y_4)$	x_3
a_5		x_3
a_6		x_3
a_3	$a_4\ (y_5 y_6 y_7)$	x_4
a_2	$a_5\ (y_8)$	$\bar{x}_3 x_1$
a_3		$\bar{x}_4 x_1$
a_4		$\bar{x}_5$
a_5		$\bar{x}_3 x_1$
a_6		$\bar{x}_3 x_1$
a_1	$a_6\ (y_1 y_3)$	$\bar{x}_1 x_2$
a_2	$a_7\ (y_3 y_4)$	$\bar{x}_3 \bar{x}_1$
a_3		$\bar{x}_4 \bar{x}_1$
a_5		$\bar{x}_3 \bar{x}_1$
a_6		$\bar{x}_3 \bar{x}_1$
a_8		1
a_1	$a_8\ (y_6 y_7)$	$\bar{x}_1 \bar{x}_2$

Après avoir établi le SGA marqué, on passe à la construction du graphe de l'automate de Moore S, dont les états sont représentés par les marques $a_1, \ldots, a_M$ obtenues dans la phase précédente. Les transitions et les signaux de sortie de l'automate seront définis comme suit.

A chaque chemin de transition $a_m X\ (a_m,\ a_s)\ a_s$ $(a_m,\ a_s \in \{a_1, \ldots, a_M\})$, on fait correspondre, dans S, la transition de l'état a_m à l'état a_s provoquée par le signal d'entrée $X\ (a_m,\ a_s)$; au chemin

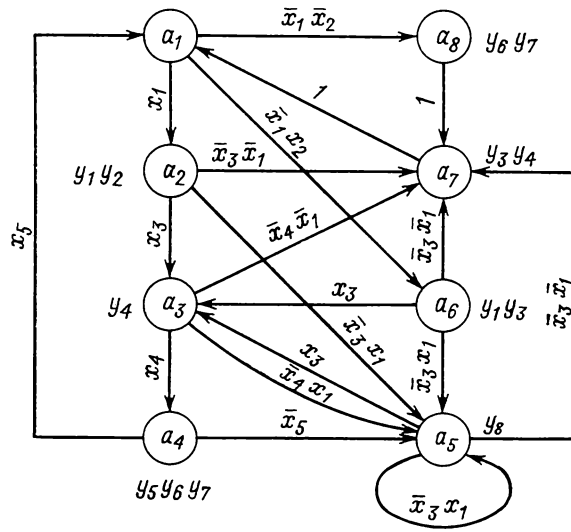


Fig. 8-6. Graphe de l'automate de Moore correspondant au SGA de la figure 8-5

de transition $a_m a_s$ correspondra la transition de a_m à a_s induite par le signal 1. En ce qui concerne les signaux de sortie, quel que soit l'état précédent, chaque transition est accompagnée par la production du signal de sortie qui est inscrit dans le sommet opératoriel marqué par cet état.

Le graphe de l'automate de Moore montré sur la figure 8-6 est construit d'après le SGA marqué de la figure 8-5. La remarque relative à la synchronisation des signaux d'entrée dans l'automate de Mealy (en fin du § 8-1) reste valable pour les automates de Moore.

Si l'automate possède un grand nombre d'états et de transitions, le graphe devient moins « parlant » : on préfère définir l'automate à l'aide d'une liste. Introduisons la notion de *table des transitions de l'automate microprogrammé*. Le tableau 8-1 est la table des transitions de l'automate de Mealy synthétisé d'après le SGA de la figure 8-1. Ce tableau a quatre colonnes : a_m, état initial ; a_s, état

de transition ; $X(a_m, a_s)$, signal d'entrée ayant pour effet la transition (a_m, a_s) ; $Y(a_m, a_s)$, signal de sortie produit lors de la transition (a_m, a_s).

A chaque chemin de transition correspond donc une ligne de la table des transitions. Si la table énumère successivement toutes les transitions à partir du premier état, puis toutes les transitions à partir du deuxième état, etc. (voir tableau 8-1), nous dirons que la table des transitions est *directe*. Dans bien des cas, il est bon d'établir la table *inverse* qui, dans ses quatre colonnes analogues à la table précédente, énumère d'abord toutes les transitions vers le premier état, puis toutes les transitions vers le deuxième état, et ainsi de suite.

S'il s'agit d'un automate de Moore, trois colonnes suffisent : dans la table directe, on inscrit le signal de sortie à côté de l'état initial qui lui correspond, et dans la table inverse, à côté de l'état vers lequel il y a transition. Le tableau 8-2 est la table des transitions inverse de l'automate de Moore construit d'après le SGA de la figure 8-5.

Il est évident qu'on a intérêt à construire la table des transitions (directe ou inverse) de l'automate microprogrammé directement d'après le SGA marqué en y inscrivant tous les chemins de transition ; autrement dit, il devient inutile de tracer le graphe, puisque la table est précisément le graphe de l'automate présenté sous forme d'une liste.

8-3. Minimisation des automates microprogrammés

Les algorithmes de synthèse des automates de Mealy d'après les SGA décrits plus haut permettent de réaliser des automates assez proches d'automates minimaux, grâce au fait que l'entrée du sommet postérieur au sommet opératoriel ne reçoit qu'une marque unique, sans rapport avec les chemins de transition aboutissant au sommet donné. L'automate sera plus ou moins proche du minimal, selon que le schéma-graphe reflète plus ou moins fidèlement le fonctionnement du système. D'autre part, on a la possibilité de minimiser les automates microprogrammés, en adaptant à cet effet la méthode de minimisation des automates abstraits complètement spécifiés décrite dans le troisième chapitre. Rappelons que deux états de l'automate de Mealy sont dits 1-équivalents si, placé dans ces états, l'automate produit sous l'action des signaux d'entrée identiques, les mêmes signaux de sortie. Pour savoir si deux états a_i et a_j de l'automate microprogrammé sont 1-équivalents ou non, on a donc besoin de comparer les ensembles de micro-instructions qui sont élaborées lors des transitions à partir des états considérés. Supposons que ces ensembles se confondent et sont égaux à $\{Y_1, \ldots, Y_r, \ldots, Y_R\}$, la micro-instruction Y_r étant induite par les signaux $X_i(Y_r)$ et $X_j(Y_r)$ pour

les états a_i et a_j respectivement. Les états a_i et a_j seront alors 1-équivalents si les fonctions $X_i(Y_r)$ et $X_j(Y_r)$ sont équivalentes pour tout $r = 1, \ldots, R$ (si les fonctions booléennes ne sont pas définies sous la forme normale disjonctive, on peut contrôler leur équivalence en effectuant l'opération de soustraction des recouvrements cubiques [45]).

Les états 1-équivalents ainsi trouvés fournissent une partition π_1 de l'ensemble d'états de l'automate microprogrammé en classes d'équivalence.

Pour l'automate de Mealy résumé dans le tableau 8-3, on a

$$\pi_1 = \overline{1.5},\ \overline{2.3.4},\ \overline{6.7},\ \overline{8.9} = \{B_1, B_2, B_3, B_4\}.$$

Les résultats de la partition de π_1 en classes d'équivalence sont présentés dans le tableau 8-4, qui, au lieu des états de transition, donne leurs classes d'équivalence.

Deux états a_i et a_j d'un automate microprogrammé sont k-équivalents si les conditions suivantes sont vérifiées:

1) a_i et a_j sont $(k-1)$-équivalents;

2) si $D_i = \{D_{i1}, \ldots, D_{in}, \ldots, D_{iN_i}\}$ et $D_j = \{D_{j1}, \ldots, D_{jn}, \ldots, D_{jN_j}\}$ sont des ensembles de classes d'états $(k-1)$-équivalents dans lesquels il y a transitions à partir de a_i et de a_j respectivement, on a $D_i = D_j = D$, c'est-à-dire que $N_i = N_j = N$ et $D_{in} = D_{jn} = D_n$ pour tout $n = 1, \ldots, N$;

3) si $X_i(D_n)$ et $X_j(D_n)$ sont deux fonctions booléennes telles que $X_i(D_n) = \bigvee_{a_m \in D_n} X(a_i, a_m)$, $X_j(D_n) = \bigvee_{a_m \in D_n} X(a_j, a_m)$, la fonction $X_i(D_n)$ est équivalente à la fonction $X_j(D_n)$ pour tout $D_n \in D$, où D est un ensemble décrit dans le n° 2 ci-dessus.

Cette définition permet de voir que les méthodes de minimisation sont sensiblement les mêmes pour les automates abstraits et les automates microprogrammés, à ceci près qu'au lieu de comparer les signaux d'entrée abstraits, on contrôle l'équivalence des recouvrements correspondants.

En faisant des partitions successives en classes 1-, 2-, ..., k-, $(k+1)$-équivalentes jusqu'à ce que deux partitions π_{k+1} et π_k soient identiques, on réalise la partition de l'ensemble d'états en états équivalents. Choisissant un état dans chaque classe d'équivalence, on obtient l'ensemble A' des états de l'automate microprogrammé minimal de Mealy. En choisissant le représentant d'une classe dans A', il y a intérêt à prendre les états qui occupent le moins de lignes dans la table des transitions. Pour construire la table des transitions de l'automate minimal, on élimine de la table initiale des blocs de données se rapportant aux transitions à partir des états qui ne figurent pas dans A' et l'on remplace dans la colonne « état

Tableau 8-3

Table des transitions de l'automate non minimal de Mealy

a_m	a_s	$X(a_m, a_s)$	$Y(a_m, a_s)$
a_1	a_2	$\bar{x}_1\bar{x}_2$	Y_1
	a_3	x_1x_4	Y_3
	a_4	$\bar{x}_1x_2$	Y_2
	a_5	$x_1\bar{x}_4$	Y_6
a_2	a_6	x_4x_3	Y_5
	a_8	$\bar{x}_4\bar{x}_3$	Y_4
	a_6	$\bar{x}_4x_3$	Y_5
	a_9	$x_4\bar{x}_3$	Y_4
a_3	a_6	x_3	Y_5
	a_7	$\bar{x}_3$	Y_4
a_4	a_6	x_3	Y_5
	a_8	$\bar{x}_3$	Y_4
a_5	a_2	$\bar{x}_1\bar{x}_2$	Y_1
	a_3	x_1x_4	Y_3
	a_4	$\bar{x}_1x_2$	Y_2
	a_5	$x_1\bar{x}_4$	Y_6
a_6	a_2	x_4	Y_3
	a_9	$\bar{x}_4$	Y_5
a_7	a_4	x_4	Y_3
	a_8	$\bar{x}_4$	Y_5
a_8	a_5	1	Y_1
a_9	a_1	1	Y_1

Tableau 8-4

Partition π_1 en classes d'états 1-équivalents

B_i	a_m	B_j	$X(a_m, B_j)$	$Y(a_m, B_j)$
B_1	a_1	B_2	$\bar{x}_1\bar{x}_2$	Y_1
		B_2	x_1x_4	Y_3
		B_2	$\bar{x}_1x_2$	Y_2
		B_1	$x_1\bar{x}_4$	Y_6
	a_5	B_2	$\bar{x}_1\bar{x}_2$	Y_1
		B_2	x_1x_4	Y_3
		B_2	$\bar{x}_1x_2$	Y_2
		B_1	$x_1\bar{x}_4$	Y_6
B_2	a_2	B_3	x_4x_3	Y_5
		B_4	$\bar{x}_4\bar{x}_3$	Y_4
		B_3	$\bar{x}_4x_3$	Y_5
		B_4	$x_4\bar{x}_3$	Y_4
	a_3	B_3	x_3	Y_5
		B_3	$\bar{x}_3$	Y_4
	a_4	B_3	x_3	Y_5
		B_4	$\bar{x}_3$	Y_4
B_3	a_6	B_2	x_4	Y_3
		B_4	$\bar{x}_4$	Y_5
	a_7	B_2	x_4	Y_3
		B_4	$\bar{x}_4$	Y_5
B_4	a_8	B_1	1	Y_1
	a_9	B_1	1	Y_1

Tableau 8-5

Partition π_2 en classes d'états 2-équivalents

C_i	a_m	C_j	$X(a_m, C_j)$	$Y(a_m, C_j)$
C_1	a_1	C_2	$\bar{x}_1\bar{x}_2$	Y_1
		C_3	x_1x_4	Y_3
		C_2	$\bar{x}_1x_2$	Y_2
		C_1	$x_1\bar{x}_4$	Y_6
	a_5	C_2	$\bar{x}_1\bar{x}_2$	Y_1
		C_3	x_1x_4	Y_3
		C_2	$\bar{x}_1x_2$	Y_2
		C_1	$x_1\bar{x}_4$	Y_6
C_2	a_2	C_4	x_4x_3	Y_5
		C_5	$\bar{x}_4\bar{x}_3$	Y_4
		C_4	$\bar{x}_4x_3$	Y_5
		C_5	$x_4\bar{x}_3$	Y_4
	a_4	C_4	x_3	Y_5
		C_5	$\bar{x}_3$	Y_4
C_3	a_3	C_4	x_3	Y_5
		C_4	$\bar{x}_3$	Y_4
C_4	a_6	C_2	x_4	Y_3
		C_5	$\bar{x}_4$	Y_5
	a_7	C_2	x_4	Y_3
		C_5	$\bar{x}_4$	Y_5
C_5	a_8	C_1	1	Y_1
	a_9	C_1	1	Y_1

Tableau 8-6

Table des transitions de l'automate minimal de Mealy

a_m	a_s	$X(a_m, a_s)$	$Y(a_m, a_s)$
a_1	a_4	$\bar{x}_1\bar{x}_2$	Y_1
	a_3	x_1x_4	Y_3
	a_4	$\bar{x}_1x_2$	Y_2
	a_1	$x_1\bar{x}_4$	Y_6
a_3	a_6	x_3	Y_5
	a_6	$\bar{x}_3$	Y_4
a_4	a_6	x_3	Y_5
	a_8	$\bar{x}_3$	Y_4
a_6	a_4	x_4	Y_3
	a_8	$\bar{x}_4$	Y_5
a_8	a_1	1	Y_1

de transition » les états ne figurant pas dans A' par leurs équivalents dans A'.

D'après le tableau 8-4, on obtient la partition π_2 (tableau 8-5) en classes d'états 2-équivalents :

$$\pi_2 = \overline{1.5},\ \overline{2.4},\ \overline{3},\ \overline{6.7},\ \overline{8.9} = \{C_1,\ C_2,\ C_3,\ C_4,\ C_5\}.$$

La partition suivante (π_3) se confond avec π_2, aussi π_2 est-elle la partition en classes d'états équivalents. Choisissons un état dans chaque classe d'équivalence; leur réunion nous fournit l'ensemble $A' = \{a_1,\ a_3,\ a_4,\ a_6,\ a_8\}$ des états de l'automate microprogrammé minimal de Mealy. La table des transitions de l'automate minimal (tableau 8-6) s'obtient en éliminant du tableau 8-3 les blocs de données qui se rapportent aux transitions à partir des états ne figurant pas dans A' et en remplaçant dans la colonne « état de transition » les états ne figurant pas dans A' par leurs équivalents dans A'.

Pour minimiser un automate microprogrammé de Moore, il convient d'établir d'abord la partition de l'ensemble d'états en classes de 0-équivalence, dont chacune réunira les états marqués par la même micro-instruction. Dans le reste, la procédure de minimisation est la même que pour les automates microprogrammés de Mealy.

CHAPITRE 9

SYNTHÈSE DU SCHÉMA LOGIQUE DE L'AUTOMATE MICROPROGRAMMÉ

9-1. Table de structure d'un automate microprogrammé

La *table de structure* d'un automate microprogrammé est une extension de sa table des transitions. Un exemple de table de structure inverse pour l'automate défini précédemment par sa table des transitions (tableau 8-1) est donné dans le tableau 9-1. On y remarque trois nouvelles colonnes : $K(a_m)$ et $K(a_s)$, codes de l'état initial a_m et de l'état de transition a_s, et $\widetilde{F}(a_m, a_s)$, ensemble des composantes

Tableau 9-1

Table de structure inverse de l'automate de Mealy

a_m	$K(a_m)$	a_s	$K(a_s)$	$X(a_m, a_s)$	$Y(a_m, a_s)$	$\widetilde{F}(a_m, a_s)$
a_2	010	a_1	001	$\bar{x}_3\bar{x}_1$	y_3y_4	R_2S_3
a_3	011			$\bar{x}_4\bar{x}_1$	y_3y_4	R_2
a_4	100			x_5	—	R_1S_3
a_5	101			1	y_3y_4	R_1
a_1	001	a_2	010	x_1	y_1y_2	S_2R_3
a_1	001			$\bar{x}_1x_2$	y_1y_3	S_2R_3
a_2	010			$\bar{x}_3x_1$	y_8	—
a_3	011			$\bar{x}_4x_1$	y_8	R_3
a_4	100			$\bar{x}_5$	y_8	R_1S_2
a_2	010	a_3	011	x_3	y_4	S_3
a_3	011	a_4	100	x_4	$y_5y_6y_7$	$S_1R_2R_3$
a_1	001	a_5	101	$\bar{x}_1\bar{x}_2$	y_6y_7	S_1

des fonctions d'excitation essentielles (qui font changer d'état des éléments de mémoire) produites lors de la transition (a_m, a_s) *). Pour le codage des états des automates microprogrammés, on peut utiliser la méthode du § 5-6, qui consiste à minimiser le nombre total de commutations de la mémoire pour l'ensemble des transitions de l'automate.

Il est évident que la table de structure d'un automate de Moore aura une colonne de moins si l'on inscrit le signal d'entrée, comme on le fait pour sa table des transitions, à côté de l'état initial (dans la table directe) ou à côté de l'état de transition (dans la table inverse).

Puisque la table de structure n'est autre que le graphe de l'automate microprogrammé structuré transcrit sous forme d'une liste, on peut en tirer les expressions des fonctions d'excitation et des fonctions des sorties, comme on l'a fait au § 5-4 lors de la synthèse par la méthode des graphes.

Par exemple, les expressions de R_1 et de y_3, déduites du tableau 9-1, s'écrivent **):

$$\left.\begin{aligned} R_1 &= pT_1\overline{T}_2\overline{T}_3x_5 \vee pT_1\overline{T}_2T_3 \vee pT_1\overline{T}_2\overline{T}_3\overline{x}_5, \\ y_3 &= p\overline{T}_1T_2\overline{T}_3\overline{x}_3\overline{x}_1 \vee p\overline{T}_1T_2T_3\overline{x}_4\overline{x}_1 \vee pT_1\overline{T}_2T_3 \vee p\overline{T}_1\overline{T}_2T_3\overline{x}_1x_2. \end{aligned}\right\} \quad (9\text{-}1)$$

Or, les expressions de ce type établies pour des automates microprogrammés concrets s'avèrent trop compliquées : elles forment un système comprenant des dizaines et des centaines de fonctions de dizaines et centaines de variables. Les méthodes classiques de minimisation des fonctions booléennes dans la classe des formes normales deviennent inopérantes dans ces cas. En outre, la pratique a montré que l'économie la plus sensible sur le nombre d'éléments du circuit est réalisée non pas en faisant la minimisation dans les formes normales mais en cherchant des fonctions et des sous-fonctions qui admettent une minimisation d'ensemble et en faisant appel à la décomposition des systèmes de fonctions.

La première phase de la minimisation du schéma logique d'un automate microprogrammé consiste à réunir les composantes des fonctions d'excitation et des fonctions des sorties inscrites dans les mêmes lignes de sa table de structure. C'est ainsi que la conjonction de la cinquième ligne du tableau 9-1 $(p\overline{T}_1\overline{T}_2T_3x_1)$ peut être utilisée pour construire les fonctions y_1, y_2, S_2, R_3, celle de la sixième ligne $(p\overline{T}_1\overline{T}_2T_3\overline{x}_1x_2)$ pour la construction de y_1, y_3, S_2, R_3, etc.

*) Ici et dans le texte qui suit, les éléments de mémoire utilisés pour la synthèse sont (sauf mention contraire) des bascules du type RS ; remarquons d'ailleurs que les méthodes proposées s'adaptent sans difficulté à n'importe quel type d'éléments de mémoire.

**) La signification du symbole p a été expliquée plus haut (voir § 8-1, remarque en fin du paragraphe).

D'une façon analogue, sur le schéma de la figure 5-8, l'expression $\bar{\tau}_1\bar{\tau}_2\bar{x}_1\bar{x}_2$ figurait dans les fonctions y_1, y_2, S_2. Cette seule circonstance nous permettra, en comparaison avec la construction directe d'après les expressions du type (9-1), de réduire le schéma d'autant de fois qu'il y a en moyenne de composantes des fonctions d'excitation et des fonctions des sorties dans une ligne de la table de structure.

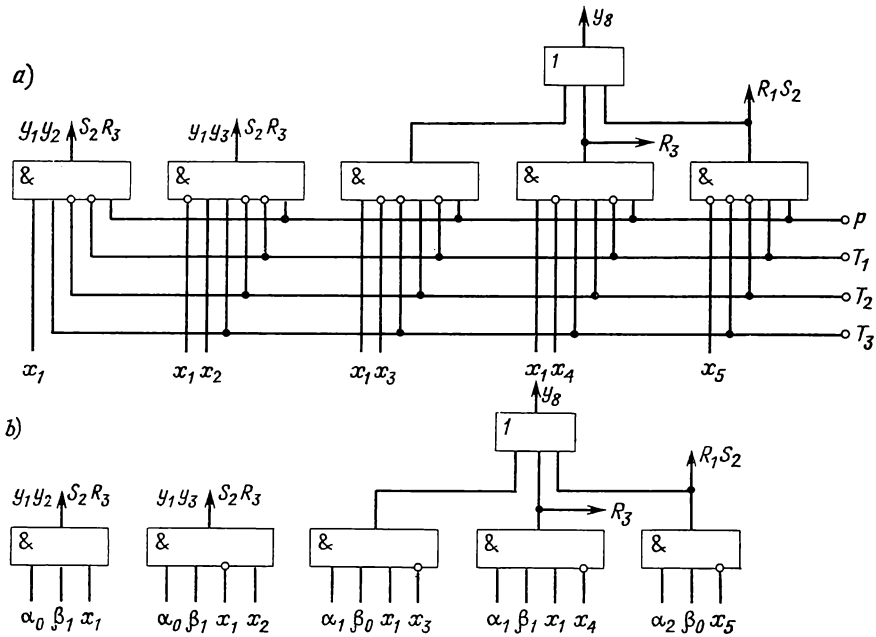


Fig. 9-1. Schéma réalisant les transitions vers l'état a_2: sans prédécodeur (*a*), avec prédécodeur (*b*)

Comme critère de minimalité (critère de coût) nous prendrons en l'occurrence le nombre total des entrées des éléments logiques du circuit.

La méthode triviale de synthèse du schéma logique d'après la table de structure inverse consiste à construire le schéma séparément pour les transitions vers chaque état; les blocs de données correspondants dans la table de structure inverse sont séparés entre eux par des lignes horizontales. Pour toutes les transitions vers un certain état, on réalise un circuit indépendant pour chaque micro-instruction. Par exemple, parmi les transitions vers l'état a_2 on a trois micro-instructions: $\{y_1, y_2\}$, $\{y_1, y_3\}$ et $\{y_8\}$. Les conjonctions correspondant à chaque ligne d'une micro-instruction sont appliquées

à un circuit OU qui produit les signaux de sortie associés à cette micro-instruction, c'est-à-dire les micro-opérations (voir la figure 9-1, *a* pour les transitions vers l'état a_2). Pour les micro-instructions $\{y_1, y_2\}$ et $\{y_1, y_3\}$ on n'a pas besoin de circuits OU, car à chacune d'elles correspond une seule ligne de la table exprimant la transition vers a_2. Si l'intersection de toutes les micro-instructions n'est pas vide, on peut prévoir encore un circuit OU qui fournira tous les signaux de sortie figurant dans l'intersection. Le coût du schéma de la figure 9-1, *a* est égal à 36 [31 entrées des circuits ET, OU plus cinq marques des fonctions d'excitation et des sorties (autres que y_1, y_2, y_8 et S_2) qu'on rencontre sur les transitions vers d'autres états et qui nécessiteront donc des circuits OU supplémentaires pour recevoir ces signaux].

9-2. Prédécodeur de retour

Les expressions (9-1) montrent que chaque composante des fonctions d'excitation et des signaux de sortie, qui correspond à un chemin de transition déterminé, se présente sous forme de conjonction des variables utilisées pour le codage du signal d'horloge p et du signal d'entrée figurant dans le chemin de transition. Au code de chaque état de l'automate on peut associer une case de la table de Karnaugh construite pour un nombre correspondant de variables; cela permet de représenter cet état sous forme de conjonction des collections d'états des bascules de la colonne et de la ligne dont l'intersection contient l'état en question.

Admettons, pour fixer les idées, que la longueur du code soit égale à 5. En regardant la table de Karnaugh à cinq variables (tableau 9-2), on dégage, dans la conjonction A_m correspondant au code $K(a_m)$ de l'état a_m, deux parties: $A_m = \underbrace{\overline{T}_1\overline{T}_2T_3}_{\alpha_1}\underbrace{T_4T_5}_{\beta'_3} = \alpha_1\beta'_3$.

Posons $\beta_i = p\beta'_i$ $(i = 0, 1, 2, 3)$, où p est l'impulsion d'horloge. Il est évident qu'en construisant les circuits qui fournissent les valeurs $\alpha_j = T_1^{e_{j1}}T_2^{e_{j2}}T_3^{e_{j3}}$ $(j = 0, 1, \ldots, 7)$ et $\beta_i = pT_4^{e_{i4}}T_5^{e_{i5}}$ $(i = 0, 1, 2, 3)$, où $e_{jk}, e_{ir} \in \{0, 1\}$; $T_k^0 = \overline{T}_k$, $T_k^1 = T_k$ $(k = 1, 2, \ldots, 5)$, on aura pour a_m du tableau 9-2 $pA_m = \alpha_1\beta_3$.

Le circuit réalisant α_j et β_i sera appelé *prédécodeur de retour*. Les indices inférieurs j, i affectant α_j, β_i sont les équivalents octaux des nombres binaires affectés à la colonne et à la ligne correspondantes de la table de Karnaugh; d'après ces indices, on reconstitue sans peine le code de l'état. Dans les cas où l'on veut indiquer seulement qu'il s'agit d'un état a_m donné, sans chercher à savoir son code, nous utiliserons la notation du type $\alpha^m\beta^m$, où les indices supérieurs affectant α et β signifient que α^m et β^m correspondent à la

Tableau 9-2

Table de Karnaugh à cinq variables

T_4T_5		$T_1T_2T_3$							
		000	001	011	010	110	111	101	100
		α_0	α_1	α_3	α_2	α_6	α_7	α_5	α_4
00	β'_0								
01	β'_1								
11	β'_3		a_m						
10	β'_2								

colonne et à la ligne de la table de Karnaugh dont l'intersection contient l'état a_m.

Le schéma de la figure 9-1, *b* est équivalent à celui de la figure 9-1, *a*, à cette différence près que ses signaux d'entrée α_i, β_j sont fournis par un prédécodeur: $\alpha_0 = \overline{T}_1\overline{T}_2$, $\alpha_1 = \overline{T}_1T_2$, $\alpha_2 = T_1\overline{T}_2$, $\beta_0 = p\overline{T}_3$, $\beta_1 = pT_3$. Un prédécodeur à quatre bascules du type *RS* est représenté sur la figure 9-2.

Supposons que l'automate passe de l'état a_m à l'état a_s sous l'action du signal d'entrée X (a_m, a_s), ce qui correspond au chemin de transition a_mX (a_m, a_s) Y (a_m, a_s) a_s sur le schéma-graphe d'algorithme. Le circuit réalisant cette transition se présentera alors comme il est montré sur la figure 9-3, *a*. Ici $\widetilde{F}(a_m, a_s)$ est l'ensemble des composantes des fonctions d'excitation essentielles produites lors de la transition (a_m, a_s); les valeurs α^m et β^m sont prélevées sur les sorties du prédécodeur.

Comparons cette configuration avec celle fournie par la méthode du décodage complet des codes des états. La même transition se réalisera alors par le circuit de la figure 9-3, *b*, où ET_m est le second étage (de sortie) du décodeur. En cas d'emploi d'un prédécodeur, à chaque chemin de transition généralisé de a_m à a_s correspondent deux entrées (sans tenir compte de X (a_m, a_s)) du circuit ET_k (fig. 9-3, *a*), ce qui fait que le coût du schéma de liaison en retour à partir des éléments de mémoire est égal dans ce cas à $V_1 = 2K$, où K est le nombre des transitions. Avec le décodeur complet, le

coût est égal à $V_2 = 2M + K$, où M est le nombre des états. De toute évidence, la méthode du prédécodeur est plus économique si $V_1 < V_2$, c'est-à-dire si $K < 2M$.

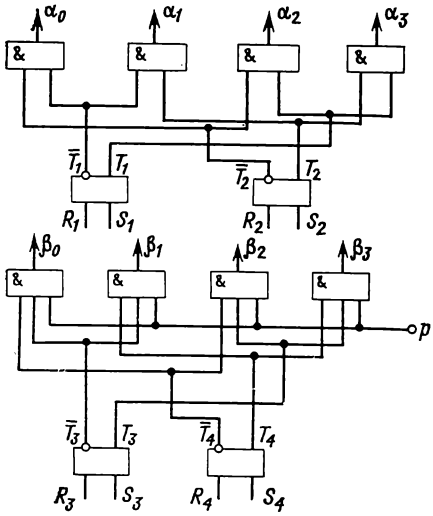


Fig. 9-2. Prédécodeur pour quatre bascules

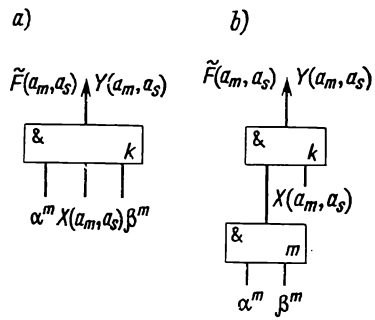


Fig. 9-3. Schémas réalisant un même chemin généralisé : avec prédécodeur (*a*), sans prédécodeur (*b*)

Au point de vue de la rapidité de fonctionnement le prédécodeur est toujours préférable au décodeur complet, car dans le dernier cas on doit prévoir un étage de plus dans le circuit.

9-3. Fonctions d'excitation complétées

Considérons le sous-graphe d'un graphe d'automate représenté sur la figure 9-4. Lors de la transition (a_i, a_k), la deuxième bascule change d'état (en passant de 0 à 1), tandis que les états des autres bascules restent inchangés ; autrement dit, l'ensemble des fonctions d'excitation essentielles pour cette transition est $\widetilde{F}(a_i, a_k) = \{S_2\}$. De même, sur la transition (a_j, a_k) on a $\widetilde{F}(a_j, a_k) = \{R_1, S_2\}$. Il est clair qu'on ne risque pas de perturber le fonctionnement de l'automate en produisant à la transition (a_i, a_k) les mêmes composantes des fonctions d'excitation qu'à la transition (a_j, a_k). En effet, si l'entrée R_1 de la première bascule se trouvant à l'état 0 reçoit le signal 1 lors de la transition (a_i, a_k), l'état de la bascule reste le même qu'auparavant.

Résumons ces raisonnements. Soit $G(a_s) = \{a_{m1}, \ldots, a_{mR}\}$ une partie de l'ensemble des états qui admettent des transitions vers l'état a_s. Les fonctions d'excitation essentielles pour ces transitions sont $\widetilde{F}(a_{mr}, a_s)$, $r = 1, \ldots, R$. *Compléter les fonctions d'excitation* lors des transitions à partir des états $a_{m1}, \ldots, a_{mR}$ vers l'état a_s consiste à produire sur toutes ces transitions un même ensemble de composantes de fonctions d'excitation:

$$\widetilde{F}(G(a_s), a_s) = \bigcup_{r=1}^{R} \widetilde{F}(a_{mr}, a_s).$$

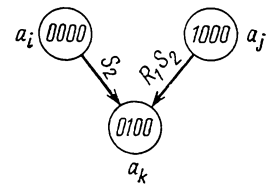


Fig. 9-4. Illustration d'une complétion des fonctions d'excitation

Autrement dit, on peut compléter les fonctions d'excitation essentielles pour les transitions considérées par des fonctions d'excitation qui laissent inchangés les états des éléments de mémoire qui le doivent être lors de la transition considérée.

Les fonctions d'excitation produites lors d'une transition déterminée (a_m, a_s) sont interprétées tantôt comme l'ensemble des signaux attaquant les entrées 1 et 0 des bascules de mémoire, tantôt comme des fonctions booléennes des signaux d'entrée de l'automate et des signaux de retour. Dans le premier cas nous utiliserons la notation $\widetilde{F}(a_m, a_s)$, et dans le second, la notation $F(a_m, a_s)$.

Par exemple, en complétant les fonctions d'excitation sur toutes les transitions vers l'état a_2 (tableau 9-1), on obtient:

$$G(a_2) = \{a_1, a_2, a_3, a_4\};$$

$$\widetilde{F}(G(a_2), a_2) = \widetilde{F}(a_1, a_2) \cup \widetilde{F}(a_2, a_2) \cup \widetilde{F}(a_3, a_2) \cup \widetilde{F}(a_4, a_2) =$$
$$= \{S_2, R_3\} \cup \varnothing \cup \{R_3\} \cup \{R_1, S_2\} = \{R_1, S_2, R_3\}.$$

L'algorithme de construction des fonctions des sorties et des fonctions de transition complétés sera examiné dans le § 9-6. Mais il nous a fallu introduire cette notion auparavant, car les fonctions d'excitation complétées sont essentiellement utilisées pour la décomposition du schéma logique à l'aide des nœuds du SGA, sujet auquel est consacré le paragraphe suivant.

9-4. Méthode de décomposition du schéma logique à l'aide des nœuds du schéma-graphe d'algorithme

Considérons le sous-graphe de SGA montré sur la figure 9-5, *a*. L'entrée de son sommet conditionnel x_{p1} marquée par Q est reliée aux sorties de deux sommets conditionnels x_{mR} et x_{nL}. On passe des

états a_m et a_n à l'état a_s par les chemins $a_m x_{m1}^{e_{m1}} \ldots x_{mR}^{e_{mR}} x_{p1}^{e_{p1}} \ldots x_{pT}^{e_{pT}} Y_g a_s$ et $a_n x_{n1}^{e_{n1}} \ldots x_{nL}^{e_{nL}} x_{p1}^{e_{p1}} \ldots x_{pT}^{e_{pT}} Y_g a_s$ respectivement (e_{mr}, e_{nl}, $e_{pt} \in \{0, 1\}$; $r = 1, \ldots, R$; $l = 1, \ldots, L$; $t = 1, \ldots, T$). Divisons la

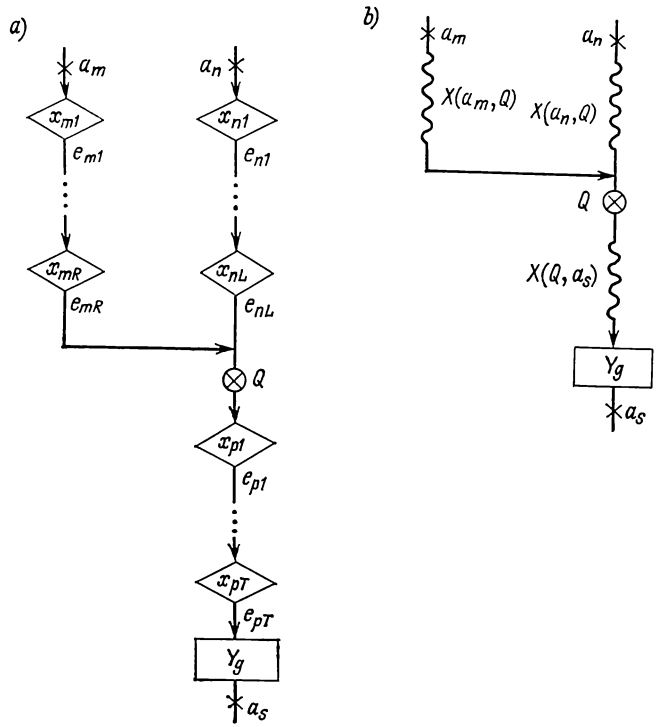


Fig. 9-5. Un nœud dans un SGA : sous-graphe comportant un nœud (*a*), représentation conventionnelle (*b*)

conjonction $X(a_m, a_s)$, qui constitue le signal d'entrée à la transition (a_m, a_s), en deux conjonctions :

$$X(a_m, a_s) = X(a_m, Q)\, X(Q, a_s).$$

De même

$$X(a_n, a_s) = X(a_n, Q)\, X(Q, a_s),$$

où

$$X(a_m, Q) = \bigwedge_{r=1}^{R} x_{mr}^{e_{mr}}, \quad X(a_n, Q) = \bigwedge_{l=1}^{L} x_{nl}^{e_{nl}}, \quad X(Q, a_s) = \bigwedge_{t=1}^{T} x_{pt}^{e_{pi}}.$$

La représentation conventionnelle du sous-graphe de la figure 9-5, *a* est donnée sur la figure 9-5, *b*. Le schéma logique traduisant les transitions (a_m, a_s) et (a_n, a_s) est représenté sur la figure 9-6, *a*.

Complétons les fonctions d'excitation sur les transitions de a_m et a_n vers a_s :

$$\widetilde{F}(a_m, a_s) \cup \widetilde{F}(a_n, a_s) = \widetilde{F}(G(Q), a_s),$$

où $G(Q) = \{a_m, a_s\}$ est l'ensemble des états à partir desquels on passe vers a_s à travers Q. Après complétion, on peut prélever l'ensem-

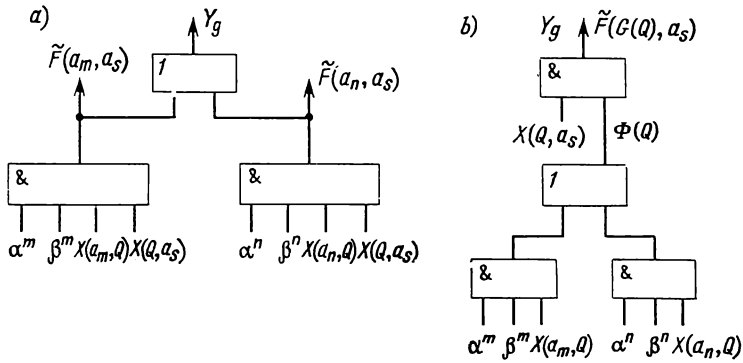


Fig. 9-6. Schéma logique pour le sous-graphe de SGA de la figure 9-5 : sans complétion des fonctions d'excitation (*a*) ; avec complétion (*b*)

ble des fonctions d'excitation $\widetilde{F}(G(Q), a_s)$ avec Y_g, ce qui permet de mettre en facteur le terme commun $X(Q, a_s)$ (fig. 9-6, *b*). Le circuit OU réalise désormais la fonction

$$\Phi(Q) = \alpha^m \beta^m X(a_m, Q) \vee \alpha^n \beta^n X(a_n, Q).$$

Considérons un nombre de chemins arbitraire issus de Q ou y aboutissant (fig. 9-7, *a*). Complétons les fonctions d'excitation sur les transitions qui conduisent de l'ensemble d'états $G(Q) = \{a_1, \ldots, a_M\}$ vers chaque état a_r $(r = M + 1, \ldots, R)$; il peut y avoir aussi $\{a_1, \ldots, a_M\} \cap \{a_{M+1}, \ldots, a_R\} \neq \varnothing$). Il vient

$$\widetilde{F}(G(Q), a_r) = \bigcup_{m=1}^{M} \widetilde{F}(a_m, a_r),$$

$$\Phi(Q) = \bigvee_{m=1}^{M} \alpha^m \beta^m X(a_m, Q),$$

$$Y_g = F(G(Q), a_r) = \Phi(Q) X(Q, a_r).$$

Le schéma logique réalisant ces expressions est représenté sur la figure 9-7, *b*.

Introduisons maintenant, complémentairement aux marques-états considérés plus haut, la notion de *marque-nœud* (ou simplement *nœud*). Nous dirons que l'entrée d'un sommet conditionnel de SGA est un nœud si elle est reliée au moins à deux sorties d'autres sommets, dont

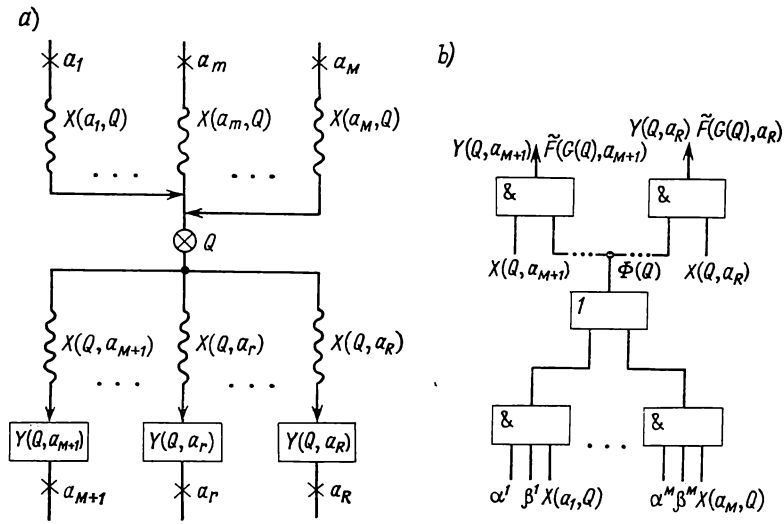


Fig. 9-7. Sous-graphe de SGA présentant plusieurs chemins d'un nœud vers les états (*a*); son schéma logique (*b*)

l'un est conditionnel. De cette façon, on peut arriver à un nœud Q_k par au moins deux chemins du type

$$a_m x_{m1}^{e_{m1}} \ldots x_{mR}^{e_{mR}} Q_k$$

partant des marques-états et qui traversent des sommets conditionnels différents $(e_{mr} \in \{0, 1\},\ x_{mr}^0 = \bar{x}_{mr},\ x_{mr}^1 = x_{mr},\ r = 1, \ldots, R)$.

Un de ces chemins peut ne pas passer par un sommet conditionnel $(R = 0)$; il s'écrit alors sous la forme $a_m Q_k$. Dans ce cas l'entrée d'un sommet conditionnel est marquée à la fois par l'état et par le nœud. Nous conviendrons de disposer la marque-nœud sous la marque-état (plus près de l'entrée du sommet, afin de ne pas perdre par mégarde le chemin qui mène de l'état considéré vers le nœud.

Dans le cas général, on peut arriver à un nœud non seulement à partir de tel ou tel état mais aussi au départ d'un autre nœud. On voit sur la figure 9-8, *a* la représentation conventionnelle d'un sous-

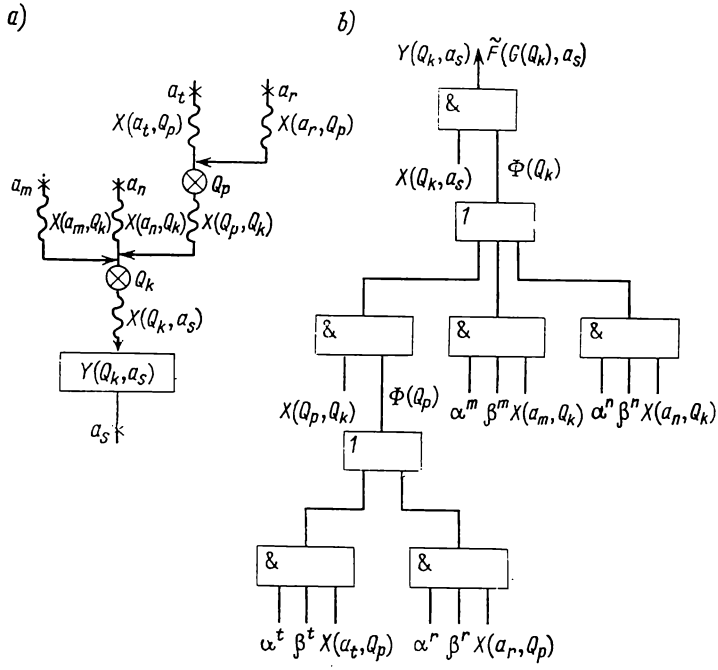


Fig. 9-8. Sous-graphe de SGA comportant deux nœuds (*a*); son schéma logique (*b*)

graphe de SGA qui comporte deux nœuds Q_k et Q_p. On a pour ce sous-graphe

$$Y(Q_k, a_s) = F(G(Q_k), a_s) = \Phi(Q_k)\, X(Q_k, a_s),$$
$$\Phi(Q_k, a_s) = \alpha^m \beta^m X(a_m, Q_k) \vee$$
$$\vee\ \alpha^n\beta^n X(a_n, Q_k) \vee \alpha^t\beta^t X(a_t, Q_k) \vee$$
$$\vee\ \alpha^r\beta^r X(a_r, Q_k).$$

Puisque $X(a_t, Q_k) = X(a_t, Q_p)\, X(Q_p, Q_k)$ et que $X(a_r, Q_k) = X(a_r, Q_p)\, X(Q_p, Q_k)$, on a

$$\Phi(Q_k) = \alpha^m\beta^m X(a_m, Q_k) \vee \alpha^n\beta^n X(a_n, Q_k) \vee$$
$$\vee\ (\alpha^t\beta^t X(a_t, Q_p) \vee \alpha^r\beta^r X(a_r, Q_p))\, X(Q_p, Q_k) =$$
$$= \alpha^m \beta^m X(a_m, Q_k) \vee \alpha^n\beta^n X(a_n, Q_k) \vee \Phi(Q_p)\, X(Q_p, Q_k),$$

où

$$\Phi(Q_p) = \alpha^t\beta^t X(a_t, Q_p) \vee \alpha^r\beta^r X(a_r, Q_p),$$

$$\widetilde{F}(G(Q_k), a_s) = \widetilde{F}(a_m, a_s) \cup \widetilde{F}(a_n, a_s) \cup \widetilde{F}(a_t, a_s) \cup \widetilde{F}(a_r, a_s) =$$
$$= \widetilde{F}(a_m, a_s) \cup \widetilde{F}(a_n, a_s) \cup \widetilde{F}(G(Q_p), a_s),$$

car $\widetilde{F}(G(Q_p, a_s)) = \widetilde{F}(a_t, a_s) \cup \widetilde{F}(a_r, a_s)$.

Nous admettrons qu'il existe un chemin allant du nœud Q_p vers le nœud Q_k s'il existe sur le schéma-graphe un chemin du type $Q_p x_{p1}^{e_{p1}} \ldots x_{pF}^{e_{pF}} Q_k$. Sur le schéma 9-8, *a* on peut arriver vers le nœud Q_k au départ des états a_m et a_n et du nœud Q_p. Dans ce cas $G(Q_k) = \{a_m, a_n\} \cup G(Q_p)$ est l'ensemble des états à partir desquels on peut arriver vers le nœud Q_k, y compris par des chemins qui traversent d'autres nœuds. Le schéma logique correspondant à la figure 9-8, *a* est donné sur la figure 9-8, *b*.

Dans le cas général, si les chemins conduisant au nœud Q_k viennent depuis les états $a_1, \ldots, a_M$ et les nœuds $Q_1, \ldots, Q_N$ et les chemins partant du nœud Q_k mènent vers les états $a_{M+1}, \ldots, a_R$ et vers les nœuds $Q_{N+1}, \ldots, Q_T$, on a

$$\Phi(Q_k) = \bigvee_{m=1}^{M} \alpha^m\beta^m X(a_m, Q_k) \vee \bigvee_{n=1}^{N} \Phi(Q_n) X(Q_n, Q_k);$$

$$Y(Q_k, a_r) = F(G(Q_k), a_r) = \Phi(Q_k) X(Q_k, a_r), \quad r = M+1, \ldots, R;$$

$$\Phi(Q_t) = \Phi(Q_k) X(Q_k, Q_t), \quad t = N+1, \ldots, T;$$

$$G(Q_k) = \{a_1, \ldots, a_M\} \cup \left(\bigcup_{n=1}^{N} G(Q_n)\right).$$

Si $G(Q_k) = \{a_{k1}, \ldots, a_{kF}\}$, on a $\widetilde{F}(G(Q_k), a_r) = \bigcup_{f=1}^{F} \widetilde{F}(a_{kf}, a_r)$.

Plaçons-nous dans le cas où il y a des chemins qui partent du nœud Q_k vers les états $a_{k1}, \ldots, a_{kR}$ et

$$\bigcap_{r=1}^{R} \widetilde{F}(G(Q_k), a_{kr}) = \widetilde{F}_k \neq \varnothing.$$

Alors, si les chemins allant de Q_k vers $a_{k1}, \ldots, a_{kR}$ ne traversent aucun sommet conditionnel à boucle sur le schéma-graphe d'algorithme, on peut recueillir à la sortie $\Phi(Q_k)$ du circuit OU l'ensemble des composantes $\widetilde{F}_k$. On a sur la figure 9-9

$$\widetilde{F}'(G(Q_k), a_{kr}) = \widetilde{F}(G(Q_k), a_{kr}) \setminus \widetilde{F}_k, \quad r = 1, \ldots, R.$$

De même, s'il y a des chemins qui partent du nœud Q_k tant vers les états $a_{k1}, \ldots, a_{kR}$ que vers les nœuds $Q_{k1}, \ldots, Q_{kN}$ et si

$$\left(\bigcap_{r=1}^{R} \widetilde{F}(G(Q_k),\ a_{kr})\right) \cap \left(\bigcap_{n=1}^{N} \widetilde{F}_{kn}\right) = \widetilde{F}_k \neq \varnothing,$$

on recueille $\widetilde{F}_k$ à la sortie $\Phi(Q_k)$; dans cette dernière expression $\widetilde{F}_{kn}$ sont les fonctions d'excitation prélevées sur le circuit OU pour $\Phi(Q_{kn})$, $n = 1, \ldots, N$.

La même considération vaut pour l'intersection des micro-instructions. Soient $Y_{k1}, \ldots, Y_{kR}$ les micro-instructions produites sur les

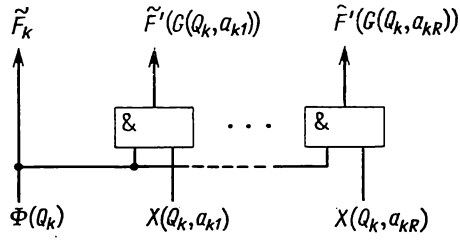


Fig. 9-9. Configuration pour le cas où $\bigcap_{r=1}^{R} \widetilde{F}(G(Q_k),\ a_{kr} = \varnothing$

transitions vers les états $a_{k1}, \ldots, a_{kR}$; supposons que les chemins de transition traversent le nœud Q_k. Alors, si $\bigcap_{r=1}^{R} Y_{kr} = Y_k \neq \varnothing$, nous prélèverons Y_k sur la sortie $\Phi(Q_k)$ du circuit OU, tout en recueillant $Y'_{kr} = Y_{kr} \setminus Y_k$ $(r = 1, \ldots, R)$ sur les circuits ET_r.

La table de structure de l'automate, introduite dans le § 9-1, présente l'automate microprogrammé sous forme d'une liste. Chaque ligne de la table de structure résume un chemin de transition, avec

Tableau 9-3

Configuration générale d'une table de structure avec des nœuds

Etat (nœud) initial	Code de l'état initial	Etat (nœud) de transition	Code de l'état de transition	Signal d'entrée	Signal de sortie	Fonctions d'excitation essentielles
a_m	$K(a_m)$	a_s	$K(a_s)$	$X(a_m, a_s)$	$Y(a_m, a_s)$	$\widetilde{F}(a_m, a_s)$
a_m	$K(a_m)$	Q_k	—	$X(a_m, Q_k)$	—	—
Q_k	—	a_s	$K(a_s)$	$X(Q_k, a_s)$	$Y(Q_k, a_s)$	$\widetilde{F}(G(Q_k), a_s)$
Q_p	—	Q_k	—	$X(Q_p, Q_k)$	—	—

Tableau 9-4

Table de structure inverse de l'automate de Mealy (avec nœuds) construite d'après la figure 9-10

Etat (nœud) initial	Code de l'état initial	Etat (nœud) de transition	Code de l'état de transition	Signal d'entrée	Signal de sortie	Fonctions d'excitation essentielles
a_2	001	Q_1	—	$\bar{x}_3\bar{x}_2$	—	—
a_4	101			$\bar{x}_8$		
Q_1	—	Q_2	—	$\bar{x}_4\bar{x}_5$	—	—
a_4	101			x_8		
a_5	010			1		
Q_2	—	Q_3	—	$\bar{x}_1$	—	—
a_1	000			$\bar{x}_1\bar{x}_6\bar{x}_4$		
a_3	011			x_6		
Q_3	—	a_1	000	$\bar{x}_7$	y_4y_9	$R_1R_2R_3$
Q_3	—			x_7	y_4y_8	$R_1R_2R_3$
Q_2	—	a_2	001	x_1	y_1y_4	$R_1R_2S_3$
a_1	000			x_1	y_1y_2	S_3
a_1	000			$\bar{x}_1x_6x_2$	y_1y_3	S_3
a_2	001			$x_3x_2\bar{x}_7$	y_1y_4	—
a_3	011			$\bar{x}_6$	y_1y_4	R_2
a_7	110			$x_3\bar{x}_4$	y_1y_4	$R_1R_2S_3$
a_2	001	a_3	011	$x_3\bar{x}_2$	y_3	S_2
a_2	001	a_4	101	$x_3x_2x_7$	y_5y_6	S_1
a_2	001			$\bar{x}_3x_2$	y_5y_6	S_1
Q_1	—	a_5	010	x_4	y_7y_8	$R_1S_2R_3$
a_1	000			$\bar{x}_1x_6\bar{x}_2$	y_7y_8	S_2
a_1	000			$\bar{x}_1\bar{x}_6x_4$	y_7y_8	S_2
a_7	110			x_3x_4	y_8	R_1
Q_1	—	a_6	100	$\bar{x}_4\bar{x}_5$	y_9	S_1R_3
a_6	100	a_7	110	1	y_1	S_2

les codes des états et les fonctions d'excitation essentielles correspondant à la transition donnée. Nous allons étendre quelque peu cette notion en ajoutant à la table de structure les chemins du type état-nœud, nœud-état et nœud-nœud. On obtient de la sorte quatre combinaisons possibles des chemins sur le schéma-graphe (voir tableau 9-3).

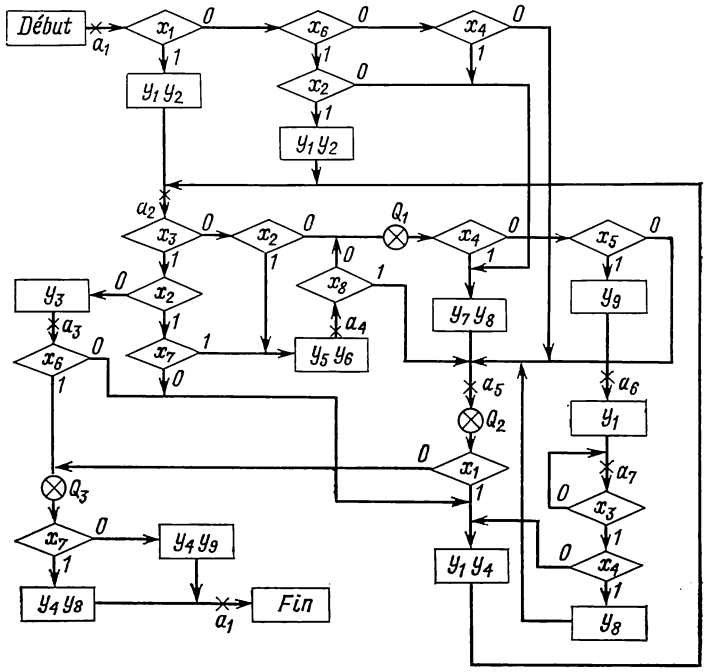


Fig. 9-10. SGA marqué

Il est à noter que le nœud n'est pas un état de l'automate; cette notion sert uniquement à faciliter la construction et la minimisation du schéma de l'automate microprogrammé; de même, la transition vers un nœud et à partir d'un nœud n'est pas une transition réelle, en ce sens qu'elle ne correspond à aucun changement d'état concret de l'automate, mais à un changement d'état « généralisé».

Cette extension acceptée, on remarque que la table de structure ne présente plus le graphe de l'automate sous forme d'une liste; par contre, elle tient déjà compte de certaines particularités de la structure du schéma à réaliser.

La table de structure inverse comportant des nœuds, construite d'après le SGA de la figure 9-10, est donnée dans le tableau 9-4. On inscrit d'abord tous les chemins (partant d'un nœud ou d'un état)

qui conduisent vers les nœuds Q_1, Q_2, . . ., puis ceux qui conduisent vers les états a_1, a_2, . . . On montre sans peine qu'en introduisant des nœuds, on arrive dans la plupart des cas à réduire le volume de la table de structure et le nombre des lettres dans le bloc « signal d'entrée » de cette table.

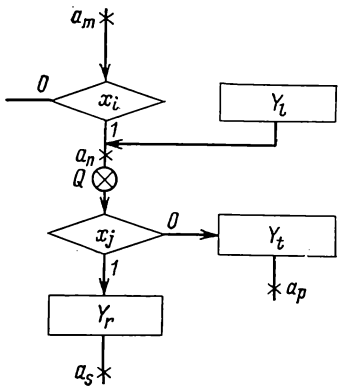


Fig. 9-11. Sous-graphe de SGA comportant un nœud

On montre en outre que l'introduction des nœuds s'avère dans tous les cas favorables vis-à-vis du coût du schéma, en entendant par coût du schéma la somme des coûts des composants, le coût d'un composant étant le nombre de ses entrées. Au lieu d'examiner le cas général, considérons le cas où l'introduction des nœuds ne présente qu'un avantage minimal. Le sous-graphe correspondant du SGA est montré sur la figure 9-11. Le schéma logique pour ce sous-graphe, dans lequel le nœud Q est absent, est donné sur la figure 9-12, *a*; on y voit quatre éléments qui totalisent quatorze entrées. Sur la figure 9-12, *b* on voit le schéma avec le nœud ; il comporte cinq éléments qui totalisent onze entrées. Proposons-nous de coder les états a_m et a_n de telle sorte

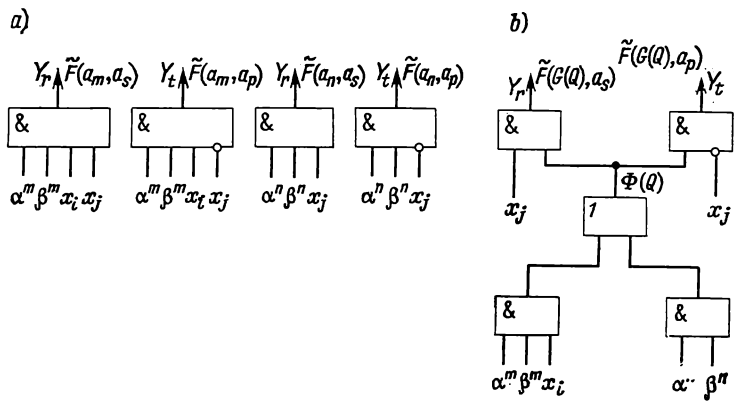


Fig. 9-12. Schéma logique sans nœud (*a*) ; avec nœud (*b*)

que $\alpha^m = \alpha^n$ ou $\beta^m = \beta^n$ (posons pour fixer les idées que $\alpha^m = \alpha^n$) ; le schéma de la figure 9-12, *b* devient celui de la figure 9-13, *a* qui compte cinq éléments et dix entrées. Introduisons à présent α^m dans les conjonctions du niveau supérieur ; le schéma prend l'aspect montré sur la figure 9-13, *b* (quatre éléments et dix entrées). En outre, si l'on

y introduit des nœuds, les signaux de sortie Y_r et Y_t et les fonctions d'excitation $\widetilde{F}$ $(G\ (Q),\ a_s)$ et $\widetilde{F}$ $(G\ (Q),\ a_p)$ sont produits une fois, alors que dans le schéma de la figure 9-12, *a* les signaux Y_r et Y_t et les fonctions d'excitation pour les transitions vers les états a_s et a_p sont produits deux fois, ce qui veut dire qu'on devra réserver à chacune de ces fonctions deux entrées de circuits OU correspondants.

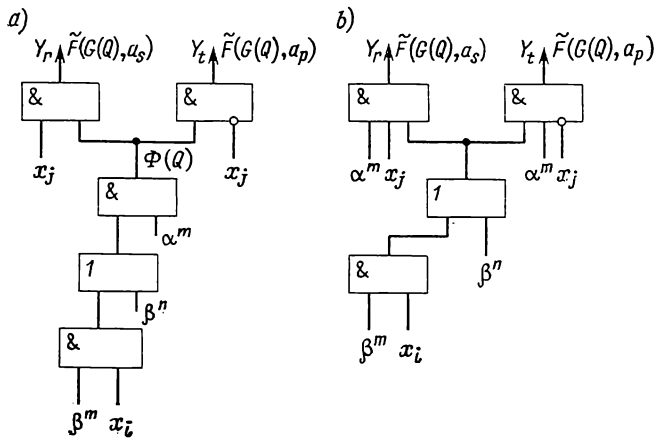


Fig. 9-13. Deux variantes du schéma logique pour $\alpha^m = \alpha^n$

Il est facile de voir que s'il existe plus de trois chemins aboutissant à un nœud (ou partant d'un nœud), un schéma faisant appel aux nœuds aura toujours moins d'éléments qu'un schéma sans nœuds.

L'algorithme de synthèse du schéma logique d'un automate microprogrammé d'après sa table de structure inverse avec des nœuds sera considéré dans le § 9-6.

9-5. Factorisation des fonctions booléennes

Soit une fonction $f = \bigvee_{t=1}^{T} X_t$ d'un ensemble de variables binaires $X = \{x_1, \ldots, x_L\}$, où $X_t = \bigwedge_{r=1}^{R_t} x_{tr}$, $t = 1, \ldots, T$; $x_{tr} \in X$. La factorisation de f se fera pas-à-pas. Sur le premier pas, en faisant toutes les intersections deux à deux sur l'ensemble $A = \{\widetilde{X}_1, \ldots, \widetilde{X}_T\}$, on cherche les parties communes ($\widetilde{X}_t = \{x_{t1}, \ldots, x_{tR_t}\}$, $t = 1, \ldots$

. . ., T). Si l'intersection de certains ensembles (par exemple, des n premiers) $\widetilde{X}_1, \ldots, \widetilde{X}_n$ n'est pas vide, en sorte que

$$\bigcap_{j=1}^{n} \widetilde{X}_j = \widetilde{Z} \neq \varnothing,$$

où $\widetilde{Z} = \{z_1, \ldots, z_m\}$, $z_i \in X$, $i = 1, \ldots, m$, on peut mettre $Z = \bigwedge_{i=1}^{m} z_i$ en facteur:

$$f = Z\,(\bigvee_{j=1}^{n} X'_j) \vee (\bigvee_{t=n+1}^{T} X_t),$$

où $\widetilde{X}'_j = \widetilde{X}_j \setminus \widetilde{Z}$.

Le gain réalisé, c'est-à-dire la diminution du coût du schéma obtenue par factorisation (mise en facteur de m lettres dans n conjonctions), est égal à $W(Z) = m(n-1) - 2 + r$ pour $n < T$ et à $W(Z) = m(n-1) - 1 + r$ pour $n = T$; ici r est le nombre de conjonctions initiales dont chacune conserve une lettre après les factorisations consécutives, et $W(Z)$, le coût de la partie commune Z. Dans l'algorithme de factorisation d'une fonction isolée la recherche des parties communes est suivie de la mise en facteur de la partie qui présente le coût le plus élevé. Une première factorisation de A conduit à deux ensembles $A_0 = \{\widetilde{X}'_1, \ldots, \widetilde{X}'_n\}$ et $B_0 = \{\widetilde{Z}, X_{n+1}, \ldots, X_T\}$, qui sont ensuite traités séparément au deuxième pas. La procédure se termine quand on constate, à un certain pas, qu'il ne reste aucune partie commune d'un coût supérieur à zéro.

A la différence de la factorisation d'un système de fonctions que nous examinerons un peu plus tard, le problème de mise en facteur sera appelé *factorisation vers le haut*, par allusion au fait que le facteur commun s'écrit au-dessus des circuits OU. On ne saurait affirmer que cet algorithme conduira nécessairement à un schéma optimal comme d'ailleurs l'algorithme proposé dans [45], où l'on propose de mettre en facteur à chaque pas la partie commune de plus grande longueur. Une grande expérience accumulée dans les transformations des circuits combinationnels d'automates microprogrammés permet d'en venir à la conclusion que la mise en facteur suivant le coût maximal procure dans la plupart des cas une solution proche de l'optimale. Pour la fonction $f = X_1 \vee X_2 \vee X_3 \vee X_4 \vee X_5$, où $X_1 = x_1x_2x_3x_4x_5x_6x_7x_{11}$, $X_2 = x_1x_2x_3x_8$, $X_3 = x_1x_2x_5x_6x_{10}x_{11}x_{12}$, $X_4 = x_5x_6x_9$ et $X_5 = x_1x_2x_5x_6x_{10}x_{12}x_{13}$, la procédure de mise en facteur selon cet algorithme est illustrée sur la figure 9-14, *a*, *b*.

Donnons quelques commentaires à l'exemple examiné *). Cher-

*) Dans l'exemple qui suit, on remplace les variables (x_i) par leurs indices (i).

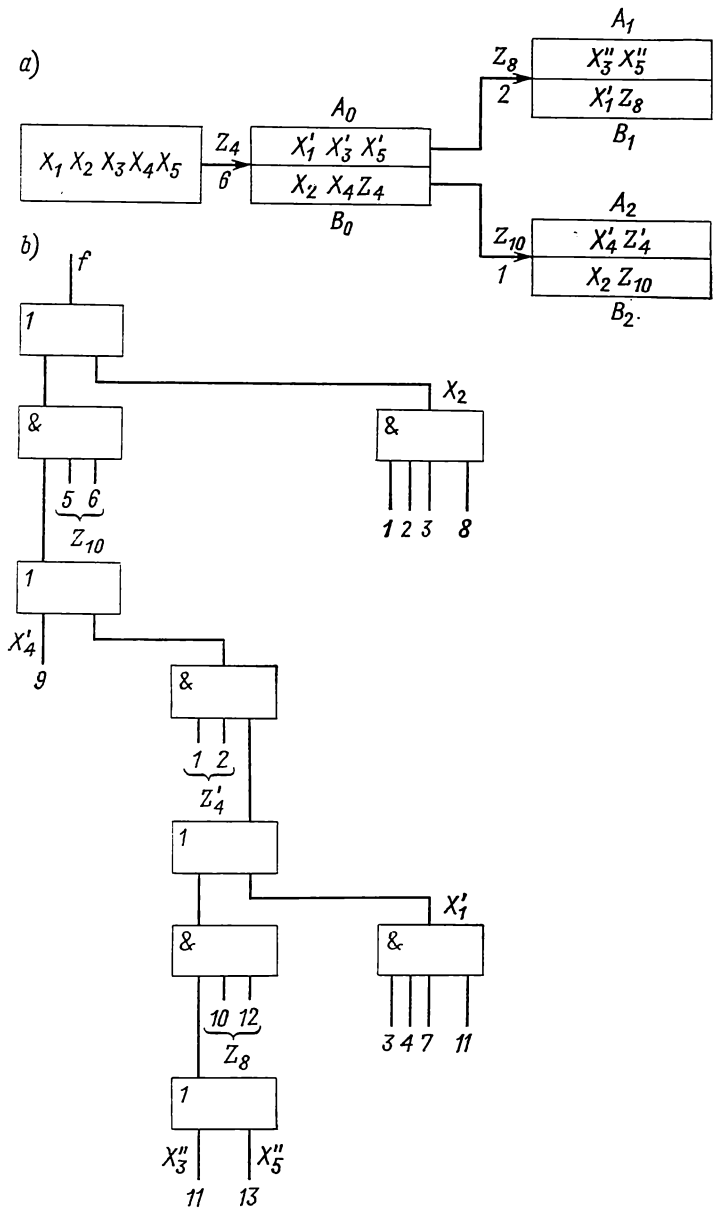


Fig. 9-14. Exemple de mise en facteur vers le haut : représentation conventionnelle de la procédure (*a*), schéma logique (*b*)

chons les intersections deux à deux dans l'ensemble initial A :

$X_1 = 1, 2\ 3, 4, 5, 6, 7, 11$	X_1			
$X_2 = 1, 2, 3, 8$	$1, 2, 3$	X_2		
$X_3 = 1, 2, 5, 6, 10, 11, 12$	$1, 2, 5, 6, 11$	$1, 2$	X_3	
$X_4 = 5, 6, 9$	$5, 6$	—	$5, 6$	X_4
$X_5 = 1, 2, 5, 6, 10, 12, 13$	$1, 2, 5, 6$	$1, 2$	$1, 2, 5, 6, 10, 12$	$5, 6$

Nous obtenons les parties communes

$$Z_1 = 1, 2, 3\,(X_1, X_2), \qquad Z_4 = 1, 2, 5, 6\,(X_1, X_3, X_5),$$
$$Z_2 = 1, 2, 5, 6, 11\,(X_1, X_3), \qquad Z_5 = 1, 2\ \ (X_1, X_2, X_3, X_5),$$
$$Z_3 = 5, 6\,(X_1, X_3, X_4, X_5), \qquad Z_6 = 1, 2, 5, 6, 10, 12\,(X_3, X_5).$$

Après chaque partie commune, on a indiqué entre parenthèses les conjonctions dont elle provient. Le gain réalisé par la mise en facteur de chaque partie commune est respectivement :

$$W(Z_1) = 2, \quad W(Z_2) = 3, \quad W(Z_3) = 5,$$
$$W(Z_4) = 6, \quad W(Z_5) = 4, \quad W(Z_6) = 6.$$

Sur le premier pas, on choisit pour la mise en facteur la partie commune Z_4 qui a le coût le plus élevé. L'ensemble initial $A = \{X_1, X_2, X_3, X_4, X_5\}$ se scinde en deux ensembles : $A_0 = \{X_1', X_3', X_5'\}$, où $\widetilde{X}_1' = \widetilde{X}_1 \setminus \widetilde{Z}_4$, $\widetilde{X}_3' = \widetilde{X}_3 \setminus \widetilde{Z}_4$, $\widetilde{X}_5' = \widetilde{X}_5 \setminus \widetilde{Z}_4$, et $B_0 = \{X_2, X_4, Z_4\}$.

Cherchons les intersections dans l'ensemble A_0 :

$X_1' = 3, 4, 7, 11$	X_{11}'	
$X_3' = 10, 11, 12$	11	X_3'
$X_5' = 10, 12, 13$	—	$10, 12$

$$Z_7 = 11\,(X_1', X_3'), \qquad W(Z_7) = -1,$$
$$Z_8 = 10, 12\,(X_3', X_5'), \quad W(Z_8) = 2.$$

Mettant en facteur Z_8, on obtient deux ensembles : $A_1 = \{X_3'', X_5''\}$, où $\widetilde{X}_3'' = \widetilde{X}_3' \setminus \widetilde{Z}$, $\widetilde{X}_5'' = \widetilde{X}_5' \setminus Z_8$, et $B_1 = \{X_1', Z_8\}$.

Dans chacun des ensembles A_1, B_1 les conjonctions qui restent sont disjointes. Aucune mise en facteur ne peut donc être faite.

Passons à l'ensemble B_0 :

$X_2 = 1, 2, 3, 8$	X_2	
$X_4 = 5, 6, 9$	—	X_4
$Z_4 = 1, 2, 5, 6$	$1, 2$	$5, 6$

$$Z_9 = 1, 2\,(X_2, Z_4), \qquad W(Z_9) = 0,$$
$$Z_{10} = 5, 6\,(X_4, Z_4), \quad W(Z_{10}) = 1.$$

La mise en facteur de Z_{10} donne lieu à deux ensembles: $A_2 = \{X_4', Z_4'\}$, où $\widetilde{X}_4' = \widetilde{X}_4 \setminus \widetilde{Z}_{10}$, $\widetilde{Z}_4' = \widetilde{Z}_4 \setminus \widetilde{Z}_{10}$, et $B_2 = \{X_2, Z_{10}\}$. Dans chacun des ensembles A_2, B_2, les conjonctions sont disjointes; aucune mise en facteur n'est donc possible. Le gain réalisé à la suite de toutes les mises en facteur s'élève à $W = W(Z_4) + W(Z_8) + W(Z_{10}) = 9$.

Les expériences d'application de l'algorithme de factorisation vers le haut réalisées par l'auteur montrent que l'insertion de cet algorithme dans un système de programmes d'automatisation de la synthèse des automates microprogrammés procure un gain de 12 % en moyenne, même après avoir fait appel au prédécodeur et à la méthode des nœuds.

La factorisation d'un système de fonctions booléennes (factorisation vers le bas) se réduit également à des mises en facteur successives, mais la procédure s'opère dans un ensemble de fonctions $f_1, \ldots, f_T$ d'un ensemble de variables binaires $X = \{x_1, \ldots, x_L\}$, où $f_t = x_{t1} \odot \ldots \odot x_{tR_t}$; $t = 1, \ldots, T$. Ici $\odot$ désigne soit une disjonction, soit une conjonction, mais jamais les deux à la fois. Dans un premier pas on cherche dans l'ensemble $A = \{\widetilde{X}_1, \ldots, \widetilde{X}_T\}$ les parties communes ($\widetilde{X}_t = \{x_{t1}, \ldots, x_{tR_t}\}$). Si l'intersection de certains ensembles, par exemple des n premiers ensembles $\widetilde{X}_1, \ldots, \widetilde{X}_n$, n'est pas vide,

$$\bigcap_{j=1}^{n} \widetilde{X}_j = \widetilde{Z} \neq \varnothing,$$

où $\widetilde{Z} = \{z_1, \ldots, z_m\}$; $z_i \in X$; $i = 1, \ldots, m$, l'ensemble des fonctions peut être réalisé sous forme d'une décomposition telle que $f_j = f(\widetilde{X}_j', f_z)$, où $\widetilde{X}_j' = \varnothing$ pour $\widetilde{X}_j = \widetilde{Z}$ ou $\widetilde{X}_j' = (\widetilde{X}_j \setminus \widetilde{Z}) \cup \{x_p\}$ pour $\widetilde{X}_j \neq \widetilde{Z}$ ($x_p \notin X$; $f_z = z_1 \odot \ldots \odot z_m$; $j = 1, \ldots, n$).

Le gain (diminution du coût du schéma) réalisé par la mise en facteur de la partie commune Z à m entrées est égal à $W(Z) = m(n-1) - n + r$, où r est le nombre des ensembles parmi $\widetilde{X}_1, \ldots, \widetilde{X}_n$ qui se confondent complètement avec $\widetilde{Z}$, car les fonctions qui leur correspondent peuvent être prélevées directement à la sortie du schéma pour la partie commune Z. Conformément à l'algorithme de factorisation du système de fonctions, la recherche des parties communes dans A est suivie de la mise en facteur de la partie commune de plus grand coût: on obtient alors un nouvel ensemble $A_0 = \{\widetilde{X}_1', \ldots, \widetilde{X}_n', \widetilde{X}_{n+1}, \ldots, \widetilde{X}_T, \widetilde{Z}\}$. Ceci fait, on applique la même procédure à l'ensemble A_0, et ainsi de suite, jusqu'à ce que, à un certain pas, on constate l'absence de toute partie commune d'un coût supérieur à zéro. A la suite des mises en facteur successives de

$Z_0, Z_1, \ldots, Z_R$ de gains respectifs $W(Z_0), W(Z_1), \ldots, W(Z_R)$, on réalise un gain total qui s'élève à $W = \sum_{r=0}^{R} W(Z_r)$.

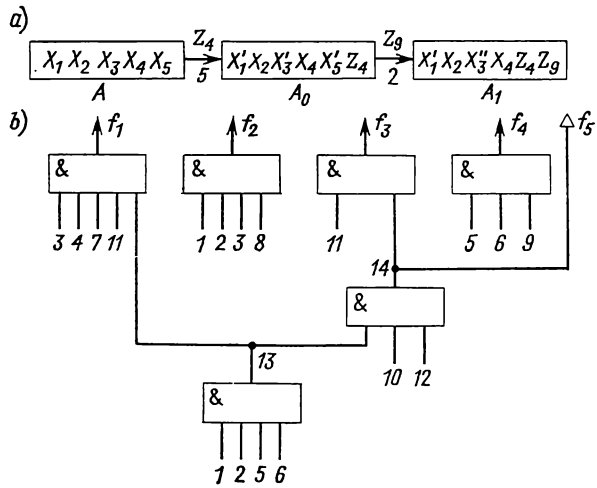


Fig. 9-15. Exemple de mise en facteur vers le bas : représentation conventionnelle de la procédure (*a*), schéma logique (*b*)

La procédure de mise en facteur selon cet algorithme pour le système de fonctions

$$
\begin{aligned}
f_1 &= x_1x_2x_3x_4x_5x_6x_7x_{11},\\
f_2 &= x_1x_2x_3x_8,\\
f_3 &= x_1x_2x_5x_6x_{10}x_{11}x_{12},\\
f_4 &= x_5x_6x_9,\\
f_5 &= x_1x_2x_5x_6x_{10}x_{12}
\end{aligned}
$$

est illustrée sur la figure 9-15, *a*, *b*. Donnons un bref commentaire à cet exemple.

Cherchons les intersections deux à deux dans l'ensemble initial $A = \{\tilde{X}_1, \tilde{X}_2, \tilde{X}_3, \tilde{X}_4, \tilde{X}_5\}$:

$X_1 = 1, 2, 3, 4, 5, 6, 7, 11$	X_1			
$X_2 = 1, 2, 3, 8$	1, 2, 3	X_2		
$X_3 = 1, 2, 5, 6, 10, 11, 12$	1, 2, 5, 6, 11	1, 2	X_3	
$X_4 = 5, 6, 9$	5, 6	—	5, 6	X_4
$X_5 = 1, 2, 5, 6, 10, 12$	1, 2, 5, 6	1, 2	1, 2, 5, 6, 10, 12	5, 6

$Z_1 = 1, 2, 3\ (X_1, X_2)$, $Z_4 = 1, 2, 5, 6\ (X_1, X_3, X_5)$,
$Z_2 = 1, 2, 5, 6, 11\ (X_1, X_3)$, $Z_5 = 1, 2\ (X_1, X_2, X_3, X_5)$,
$Z_3 = 5, 6\ (X_1, X_3, X_4, X_5)$, $Z_6 = 1, 2, 5, 6, 10, 12\ (X_3, X_5)$.

Le gain réalisé par la mise en facteur de chaque partie commune est égal respectivement à

$$W(Z_1) = 1, \quad W(Z_2) = 3, \quad W(Z_3) = 2,$$
$$W(Z_4) = 5, \quad W(Z_5) = 2, \quad W(Z_6) = 5.$$

Dans le premier pas, nous retenons pour la mise en facteur la partie commune Z_4 qui a le coût le plus élevé. Cela nous donne, au départ de l'ensemble initial, l'ensemble

$$A_0 = \{\tilde{X}'_1, \tilde{X}_2, \tilde{X}'_3, \tilde{X}_4, \tilde{X}'_5, \tilde{Z}_4\},$$
$$\tilde{X}'_1 = (\tilde{X}_1 \setminus \tilde{Z}_4) \cup \{13\}, \quad \tilde{X}'_3 = (\tilde{X}_3 \setminus \tilde{Z}_4) \cup \{13\}, \quad \tilde{X}'_5 = (\tilde{X}_5 \setminus \tilde{Z}_4) \cup \{13\}.$$

Cherchons à présent dans A_0 les intersections deux à deux qui contiennent au moins deux lettres:

$X'_1 = 3, 4, 7, 11, 13$	X'_1				
$X_2 = 1, 2, 3, 8$	—	X_2			
$X'_3 = 10, 11, 12, 13$	11, 13	—	X'_3		
$X_4 = 5, 6, 9$	—	—	—	X_4	
$X'_5 = 10, 12, 13$	—	—	10, 12, 13	—	X'_5
$Z_4 = 1, 2, 5, 6$	—	1, 2	—	5, 6	—

$$Z_7 = 11, 13 \ (X'_1, X'_3), \qquad W(Z_7) = 0,$$
$$Z_8 = 1, 2 \ (X_2, Z_4), \qquad W(Z_8) = 0,$$
$$Z_9 = 10, 12, 13 \ (X'_3, X'_5), \qquad W(Z_9) = 2,$$
$$Z_{10} = 5, 6 \ (X_4, Z_4), \qquad W(Z_{10}) = 0.$$

La mise en facteur de la partie commune $Z_9 = 10, 12, 13$ nous donne un nouvel ensemble A_1 qui n'a aucune intersection de plus d'une lettre. Le gain réalisé à la suite des mises en facteur est $W = W(Z_4) + W(Z_9) = 5 + 2 = 7$.

Lors de la synthèse d'un automate microprogrammé d'après son SGA, après avoir mis en œuvre la méthode des nœuds (recherche des transitions qui aboutissent à des nœuds et à des états isolés), on peut poursuivre la minimisation du schéma en faisant les factorisations vers le haut (pour chaque fonction isolée) et vers le bas (recherche des parties communes parmi les circuits ET pour des fonctions différentes). On donne à titre d'exemple, sur la figure 9-16, *a*, le schéma construit d'après le tableau 9-5 qui représente un fragment de table de structure (le coût du schéma est 70), et sur la figure 9-16, *b*, le même schéma après avoir complété toutes les fonctions d'excitation et effectué la minimisation par les factorisations indiquées (le coût du schéma est 50).

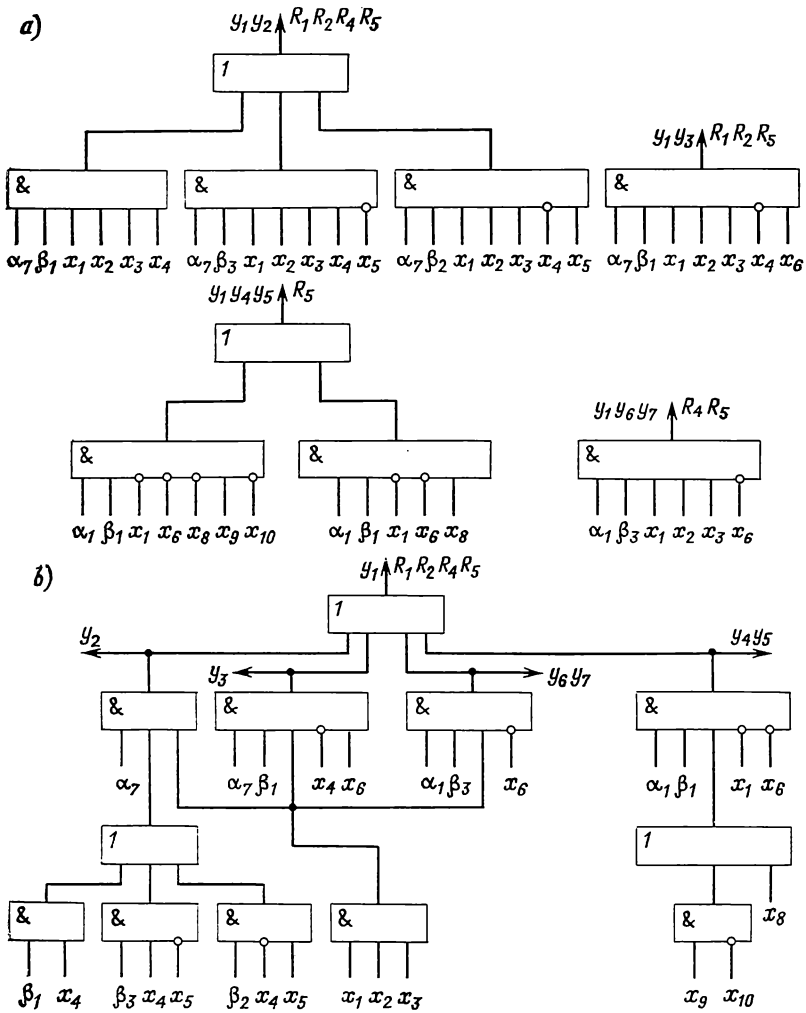


Fig. 9-16. Schéma logique construit d'après le tableau 9-5 : sans mise en facteur (*a*) ; avec mise en facteur (*б*)

Tableau 9-5

Fragment de table de structure inverse

Etat initial	Code de l'état initial	Etat de transition	Code de l'état de transition	Signal d'entrée	Signal de sortie	Fonctions d'excitation essentielles
a_1	11101	a_7	00100	$x_1x_2x_3x_4$	y_1y_2	$R_1R_2R_5$
a_1	11101			$x_1x_2x_3\bar{x}_4x_6$	y_1y_3	$R_1R_2R_5$
a_2	11111			$x_1x_2x_3x_4\bar{x}_5$	y_1y_2	$R_1R_2R_4R_5$
a_3	00111			$x_1x_2x_3\bar{x}_6$	$y_1y_6y_7$	R_4R_5
a_4	00101			$\bar{x}_1\bar{x}_6\bar{x}_8x_9\bar{x}_{10}$	$y_1y_4y_5$	R_5
a_4	00101			$\bar{x}_1\bar{x}_6x_8$	$y_1y_4y_5$	R_5
a_6	11110			$x_1x_2x_3\bar{x}_4x_5$	y_1y_2	$R_1R_2R_4$

En plus des circuits ET, l'opération de factorisation vers le bas peut s'appliquer aux circuits OU, fournissant les signaux de sortie ou les composantes des fonctions d'excitation, par exemple. Nous le montrerons sur l'exemple du tableau 9-6 qui est un fragment de table de structure d'un automate. Puisque, dans le cas considéré, la forme concrète des signaux d'entrée n'a aucune importance, nous les avons mis sous la forme $X_1, \ldots, X_7$.

Il est évident qu'en construisant un schéma pour chaque ligne du tableau 9-6, nous obtiendrons sept circuits ET qui produisent les

Tableau 9-6

Fragment de table de structure

Etat initial	Code de l'état initial	Etat de transition	Code de l'état de transition	Signal d'entrée	Signal de sortie	Fonctions d'excitation essentielles
a_{t1}	1010010	a_t	1010000	X_1	$y_{t1}y_{t2}$	R_6
a_{t1}	1010010			X_2	$y_{t3}y_{t4}$	R_6
a_{t2}	0010011			X_3	—	$S_1R_6R_7$
a_{t3}	1010011			X_4	y_{t1}	R_6R_7
a_{t4}	0111111			X_5	$y_{t1}y_{t2}$	$S_1R_2R_4R_5R_6R_7$
a_{t4}	0111111			X_6	$y_{t1}y_{t2}$	$S_1R_2R_4R_5R_6R_7$
a_{t5}	0110110			X_7	$y_{t3}y_{t4}$	$S_1R_2R_5R_6$

signaux de sortie et les fonctions d'excitation inscrits dans la ligne donnée :

$$
\begin{aligned}
&\text{ET } 1 - y_{t1},\ y_{t2},\ R_6;\\
&\text{ET } 2 - y_{t3},\ y_{t4},\ R_6;\\
&\text{ET } 3 - S_1,\ R_6,\ R_7;\\
&\text{ET } 4 - y_{t1},\ R_6,\ R_7;\\
&\text{ET } 5 - y_{t1},\ y_{t2},\ S_1,\ R_2,\ R_4,\ R_5,\ R_6,\ R_7;\\
&\text{ET } 6 - y_{t1},\ y_{t2},\ S_1,\ R_2,\ R_4,\ R_5,\ R_6,\ R_7;\\
&\text{ET } 7 - y_{t3},\ y_{t4},\ S_1,\ R_2,\ R_5,\ R_6.
\end{aligned}
$$

Un signal de sortie ou une composante d'une fonction d'excitation sont élaborés par un circuit OU auquel il faut appliquer le signal de sortie de celui des circuits $ET_1, \ldots, ET_7$ qui fournit le signal donné ou la composante donnée de la fonction d'excitation (fig. 9-17, *a*).

Mettons les signaux de sortie et les fonctions d'excitation sous forme de mots formés des indices des circuits qui les produisent :

$$
\left.
\begin{array}{ll}
y_{t1}=1,\ 4,\ 5,\ 6; & R_2=5,\ 6,\ 7;\\
y_{t2}=1,\ 5,\ 6; & R_4=5,\ 6;\\
y_{t3}=2,\ 7; & R_5=5,\ 6,\ 7;\\
y_{t4}=2,\ 7; & R_6=1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7;\\
S_1=3,\ 5,\ 6,\ 7; & R_7=3,\ 4,\ 5,\ 6.
\end{array}
\right\} \qquad (9\text{-}2)
$$

En appliquant maintenant à l'ensemble $A = \{y_{t1}, y_{t2}, y_{t3}, y_{t4}, S_1, R_2, R_4, R_5, R_6, R_7\}$ la procédure de factorisation vers le bas et en mettant en facteur à chaque pas la partie commune de coût maximal, nous obtenons ainsi le schéma montré sur la figure 9-17, *b* dont le coût est de quinze unités plus bas que celui du schéma de la figure 9-17, *a*.

Les expériences d'application de l'algorithme de factorisation vers le bas montrent qu'en insérant cet algorithme dans le système automatisé de synthèse des automates microprogrammés, on arrive à abaisser le coût du schéma de 19 % en moyenne, même après avoir tiré tout le profit de l'emploi du prédécodeur, de la méthode des nœuds et de la procédure de factorisation vers le haut.

On peut appliquer la factorisation vers le bas à la minimisation des circuits OU pour les signaux de sortie et les fonctions d'excita-

tion, en la combinant avec la complétion des fonctions d'excitation. A cet effet, on donne à chaque signal de sortie la forme d'un mot constitué par les indices des circuits qui le produisent (comme dans

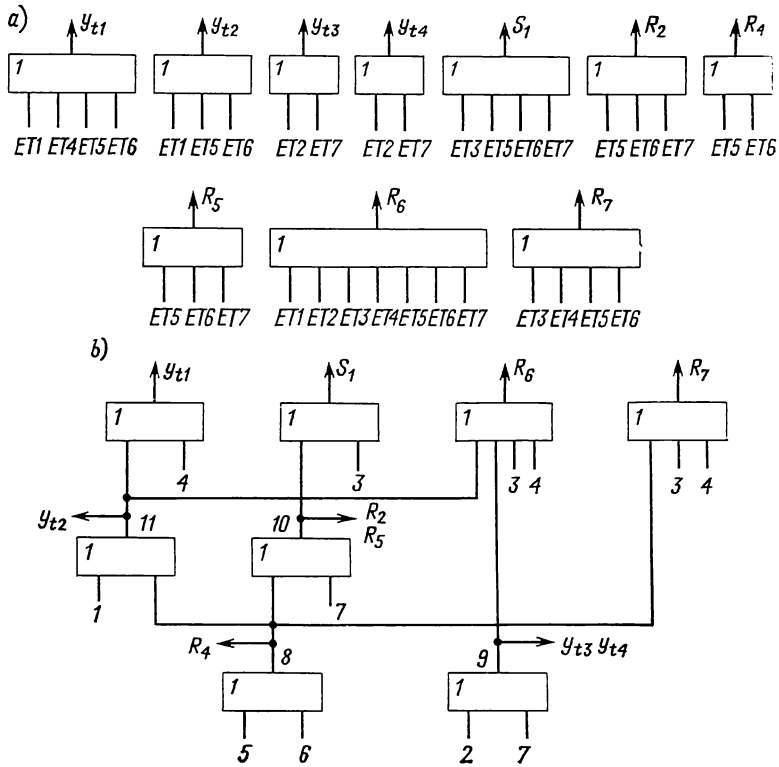


Fig. 9-17. Réalisation des signaux de sortie et des fonctions d'excitation du tableau 9-6 : sans mise en facteur vers le bas (*a*), avec mise en facteur vers le bas (*b*)

les expressions (9-2)). Pour une composante S_t (R_t) de la fonction d'excitation, un tel mot aura deux parties :

a) une partie essentielle C_t : c'est l'ensemble des indices des circuits ET qui produisent les fonctions d'excitation essentielles ;

b) une partie non essentielle D_t (on la met entre parenthèses dans l'ensemble S_t (R_t)) : c'est l'ensemble des indices des circuits ET qui produiront cette composante de S_t (R_t) après que l'on aura complété les fonctions d'excitation pour toutes les transitions.

Par exemple, après avoir complété les fonctions d'excitation pour les transitions qui mènent vers l'état a_t du tableau 9-6, on obtient

l'ensemble initial suivant:

$$A = \{y_{t1}, y_{t2}, y_{t3}, y_{t4}, S_1, R_2, R_4, R_5, R_6, R_7\};$$

$$\left.\begin{array}{ll} y_{t1}=1,\ 4,\ 5,\ 6; & R_2=5,\ 6,\ 7\ (1,\ 2,\ 3,\ 4); \\ y_{t2}=1,\ 5,\ 6; & R_4=5,\ 6\ (1,\ 2,\ 3,\ 4,\ 7); \\ y_{t3}=2,\ 7; & R_5=5,\ 6,\ 7\ (1,\ 2,\ 3,\ 4); \\ y_{t4}=2,\ 7; & R_6=1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7; \\ S_1=3,\ 5,\ 6,\ 7\ (1,\ 2,\ 4); & R_7=3,\ 4,\ 5,\ 6\ (1,\ 2,\ 7). \end{array}\right\} \quad (9\text{-}3)$$

Faisons dans A toutes les intersections deux à deux (comme dans la procédure de factorisation vers le bas examinée plus haut) sans faire la distinction entre les lettres de la partie essentielle et celles de la partie non essentielle de S_t (R_t). Nous obtiendrons les parties communes $Z_{01}, \ldots, Z_{0E_0}$. Après Z_{0e} $(e = 1, \ldots, E_0)$, nous écrivons entre parenthèses y_n si la partie commune $\widetilde{Z}_{0e}$ intervient dans y_n, et S_t^k (R_t^k) si $\widetilde{Z}_{0e}$ intervient dans S_t (R_t) et si $\widetilde{Z}_{0e} \cap C_t \neq \varnothing$, c'est-à-dire si au moins une lettre de $\widetilde{Z}_{0e}$ figure dans la partie essentielle de S_t (R_t); $k = P\,(\widetilde{Z}_{0e} \cap D_t)$ est le nombre des lettres de la partie non essentielle de S_t (R_t) contenues dans $\widetilde{Z}_{0e}$. Si par contre $\widetilde{Z}_{0e} \cap \cap\, C_t = \varnothing$, on omet d'inscrire S_t (R_t) entre parenthèses après $\widetilde{Z}_{0e}$, admettant que Z_{0e} ne figure pas dans S_t (R_t).

Le gain totalisé par la partie commune Z et obtenu grâce à l'application des opérations de factorisation vers le bas et de complétion des fonctions d'excitation se calcule d'après la formule

$$W\,(Z) = m\,(n - 1) - n + r - p \qquad (9\text{-}4)$$

dans laquelle les symboles m, n, r ont la même signification que dans l'algorithme de factorisation vers le haut sans tenir compte de l'indétermination, tandis que p est le nombre total des indices supérieurs des fonctions d'excitation qui sont écrites entre parenthèses après la partie commune.

Le signe négatif avec lequel la quantité p intervient dans (9-4) s'explique par le fait que le gain réalisé par la factorisation vers le bas n'est pris en compte que pour la partie essentielle, étant donné que le schéma initial (voir par exemple la figure 9-17, *a*) ne comporte pas d'entrées pour des fonctions d'excitation non essentielles.

Comme précédemment, nous choisirons parmi les parties communes obtenues en premier pas un Z_0 tel que $W\,(Z_0) = \max\limits_{1 \leqslant e \leqslant E_0} (W\,(Z_{0e}))$. Après la mise en facteur de Z_0, on forme un nouvel ensemble A_0 d'après les mêmes règles que dans l'algorithme de factorisation vers le bas, puis on cherche de nouveau une partie commune, Z_1,

de plus grand coût, on construit un nouvel ensemble A_1, et ainsi de suite, jusqu'à ce que chaque nouvelle partie commune ait un coût nul. Le gain total réalisé par les mises en facteur successives de Z_0, $Z_1, \ldots, Z_R$ s'élève à $\sum_{r=0}^{R} W(Z_r)$.

Donnons une illustration de cet algorithme en examinant les expressions (9-3):

$y_{t1}=1, 4, 5, 6$	y_{t1}				
$y_{t2}=1, 5, 6$	1, 5, 6	y_{t2}			
$y_{t3}=2, 7$	—	—	y_{t3}		
$y_{t4}=2, 7$	—	—	2, 7	y_{t4}	
$S_1=3, 5, 6, 7\ (1, 2, 4)$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	$S_1, R_2, R_4, R_5, R_6, R_7$
$R_2=5, 6, 7\ (1, 2, 3, 4)$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	1, 2, 3, 4, 5, 6, 7
$R_4=5, 6\ (1, 2, 3, 4, 7)$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	1, 2, 3, 4, 5, 6, 7
$R_5=5, 6, 7\ (1, 2, 3, 4)$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	1, 2, 3, 4, 5, 6, 7
$R_6=1, 2, 3, 4, 5, 6, 7$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	1, 2, 3, 4, 5, 6, 7
$R_7=3, 4, 5, 6\ (1, 2, 7)$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7	1, 2, 3, 4, 5, 6, 7

$$Z_1 = 1, 2\ 3, 4, 5, 6, 7\ (S_1^3, R_2^4, R_4^5, R_5^4, R_6^0, R_7^3);$$
$$Z_2 = 1, 4, 5, 6\ (y_{t1}, S_1^2, R_2^2, R_4^2, R_5^2, R_6^0, R_7^1);$$
$$Z_3 = 1, 5, 6\ (y_{t1}, y_{t2}, S_1^1, R_2^1, R_4^1, R_5^1, R_6^0, R_7^1);$$
$$Z_4 = 2, 7\ (y_{t3}, y_{t4}, S_1^1, R_2^1, R_5^1, R_6^0).$$

$W(Z_1) = 7\cdot 5 - 6 + 6 - 19 = 16$; $W(Z_2) = 4\cdot 6 - 7 + 1 - 9 = 9$;
$W(Z_3) = 3\cdot 7 - 8 + 1 - 5 = 9$; $W(Z_4) = 2\cdot 5 - 6 + 2 - 3 = 3$;

$W(Z_1) = \max\limits_{1\leqslant i\leqslant 4} W(Z_i) = 16$ est le coût maximal qui caractérise la partie commune Z_1; nous mettons donc Z_1 en facteur dans l'ensemble $S_1, R_2, R_4, R_5, R_6, R_7$ et nous supprimons les fonctions d'excitation énumérées de l'ensemble initial, car elles se confondent toutes avec la partie commune. La sortie du circuit réalisant Z_1 sera désignée par 8. Cherchons les intersections dans le nouvel ensemble A_0:

$y_{t1}=1, 4, 5, 6$	y_{t1}			
$y_{t2}=1, 5, 6$	1, 5, 6	y_{t2}		
$y_{t3}=2, 7$	—	—	y_{t3}	
$y_{t4}=2, 7$	—	—	2, 7	y_{t4}
$Z_1=1, 2, 3, 4, 5, 6, 7$	1, 4, 5, 6	1, 5, 6	2, 7	2, 7

$Z_5=1, 5, 6\ (y_{t1}, y_{t2}, Z_1)$; $W(Z_5)=4$;
$Z_6=1, 4, 5, 6\ (y_{t1}, Z_1)$; $W(Z_6)=3$;
$Z_7=2, 7\ (y_{t3}, y_{t4}, Z_1)$; $W(Z_7)=3$.

Mettons en facteur Z_5. La sortie du circuit OU réalisant Z_5 sera désignée par 9.

$y'_{t1}=4,\ 9$	y'_{t1}			
$y_{t3}=2,\ 7$	—	y_{t3}		
$y_{t4}=2,\ 7$	—	2, 7	y_{t4}	
$Z'_1=2,\ 3,\ 4,\ 7,\ 9$	4, 9	2, 7	2, 7	Z'_1
$Z_5=1,\ 5,\ 6$	—	—	—	—

$$Z_8=4,\ 9\ (y'_{t1},\ Z'_1);\qquad W(Z_8)=1;$$

$$Z_9=2,\ 7\ (y_{t3},\ y_{t4},\ Z'_1);\qquad W(Z_9)=3.$$

Mettons en facteur Z_9. La sortie du circuit OU réalisant Z_9 sera désignée par 10.

$y'_{t1}=4,\ 9$	y'_{t1}		
$Z''_1=3,\ 4,\ 9,\ 10$	4, 9	Z''_1	
$Z_5=1,\ 5,\ 6$	—	—	Z_5
$Z_9=2,\ 7$	—	—	—

$$Z_{10}=4,\ 9\ (y'_{t1},\ Z''_1);\qquad W(Z_{10})=1.$$

La mise en facteur de Z_{10} achève la procédure. Le schéma résultant est donné sur la figure 9-18. Le nombre des entrées des éléments

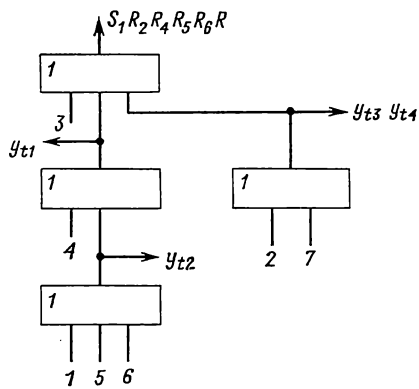


Fig. 9-18. Configuration du schéma après la mise en facteur vers le bas et complétion des fonctions d'excitation

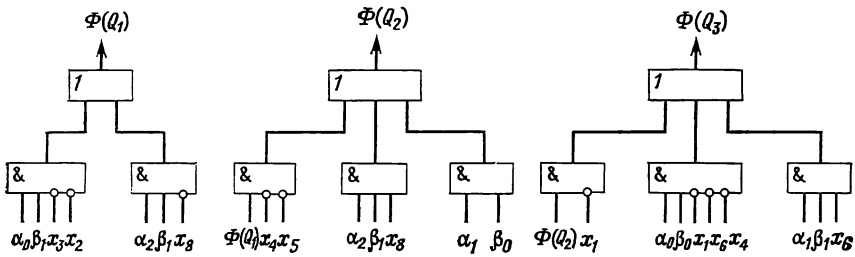
logiques sur ce schéma est de 24 inférieur à celui du schéma de la figure 9-17, *a* ($W(Z_1) + W(Z_5) + W(Z_{10}) = 24$) et de 9 inférieur à celui du schéma de la figure 9-17, *b* obtenu par l'application de la seule factorisation vers le bas, sans complétion des fonctions d'excitation.

Il est évident que la procédure décrite de factorisation vers le bas avec complétion s'applique également dans le cas où sont incomplètement spécifiés les signaux de sortie.

9-6. Synthèse du schéma logique d'un automate microprogrammé d'après sa table de structure avec des nœuds

Nous décrirons dans ce paragraphe la procédure séquentielle de construction du schéma logique d'un automate microprogrammé de Mealy, utilisant les résultats dégagés dans les paragraphes précédents.

1. Supposons que la table de structure présente des chemins aboutissant aux nœuds $Q_1, \ldots, Q_N$ et aux états $a_1, \ldots, a_M$. On divise la table en $N + M$ blocs de données dont chacun correspond à un



Fir. 9-19. Schémas logiques réalisant les fonctions $\Phi(Q_1)$, $\Phi(Q_2)$, $\Phi(Q_3)$

ensemble de chemins conduisant à un nœud ou à un état. Pour l'automate microprogrammé défini par le tableau 9-4 on a $N = 3$ et $M = 7$. Les blocs de données correspondants sont séparés entre eux par un trait horizontal.

2. Réserver à l'ensemble de chemins menant à chaque nœud Q_n ($n = 1, \ldots, N$) autant de circuits ET qu'il y a de lignes du tableau réunis dans le bloc approprié. Les circuits retenus réalisent la fonction $\Phi(Q_p)\, X(Q_p, Q_n)$ pour le chemin allant du nœud Q_p vers le nœud Q_n et la fonction $\alpha^s\beta^s X(a_s, Q_n)$ pour le chemin allant de l'état a_s vers le nœud Q_n.

3. Relier les sorties des circuits ET prévus dans le n° 2 pour le nœud Q_n au circuit OU qui produit la fonction $\Phi(Q_n)$. Les circuits élaborant les fonctions $\Phi(Q_1)$, $\Phi(Q_2)$ et $\Phi(Q_3)$ du tableau 9-4 sont représentés sur la figure 9-19.

4. Faire si possible une factorisation vers le haut pour la fonction $\Phi(Q_n)$ comme décrit dans le § 9-5.

5. Diviser chaque ensemble de chemins allant vers l'état a_m ($m = 1, \ldots, M$) en autant de sous-ensembles qu'il y a de microinstructions différentes dans la colonne « Signal de sortie » de son

bloc de données. Par exemple, aux transitions vers l'état a_5 du tableau 9-4 correspondent deux micro-instructions $\{y_7, y_8\}$ et $\{y_8\}$. Supposons que, dans le cas général, aux transitions vers l'état a_m correspondent les micro-instructions $Y_{m1}, \ldots, Y_{mT}$.

6. Réserver à chaque sous-bloc correspondant à Y_{mt} autant de circuits ET qu'il y a de lignes de la table de structure couvertes par ce sous-bloc. Ces circuits réalisent la fonction $\Phi(Q_p)\, X(Q_p, a_m)$ pour

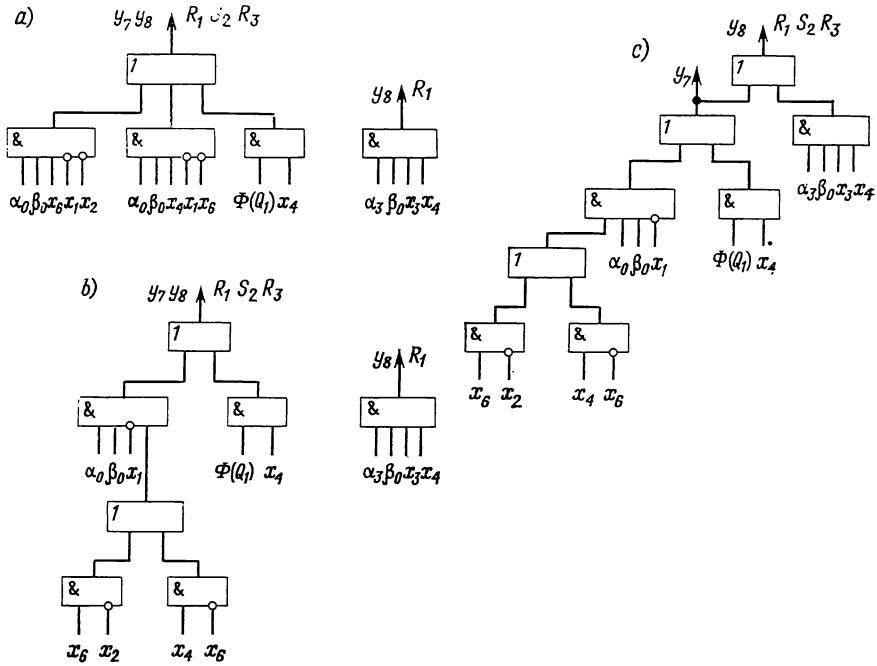


Fig. 9-20. Schémas des transitions vers a_5 : complétion des fonctions d'excitation sur les transitions de mêmes micro-instructions (*a*), mise en facteur vers le haut (*b*), complétion des fonctions d'excitation sur toutes les transitions (*c*)

le chemin allant du nœud Q_p à a_m et la fonction $\alpha^s\beta^s X(a_s, a_m)$ pour le chemin allant de l'état a_s à l'état a_m.

7. Relier les sorties des circuits ET prévus dans le n° 6 pour le sous-bloc Y_{mt} au circuit OU_{mt} qui produit les signaux des micro-opérations intervenant dans la micro-instruction Y_{mt}.

8. Compléter les fonctions d'excitation sur toutes les transitions qui correspondent aux chemins issus du sous-bloc Y_{mt}. La fonction d'excitation complétée $\widetilde{F}_{mt}$ est prélevée sur le circuit OU_{mt}. Le schéma correspondant aux transitions vers l'état a_5 du tableau 9-4 est donné sur la figure 9-20, *a*.

9. Faire si possible une factorisation vers le haut pour le circuit correspondant à la micro-instruction Y_{mt} et à la fonction d'excita-

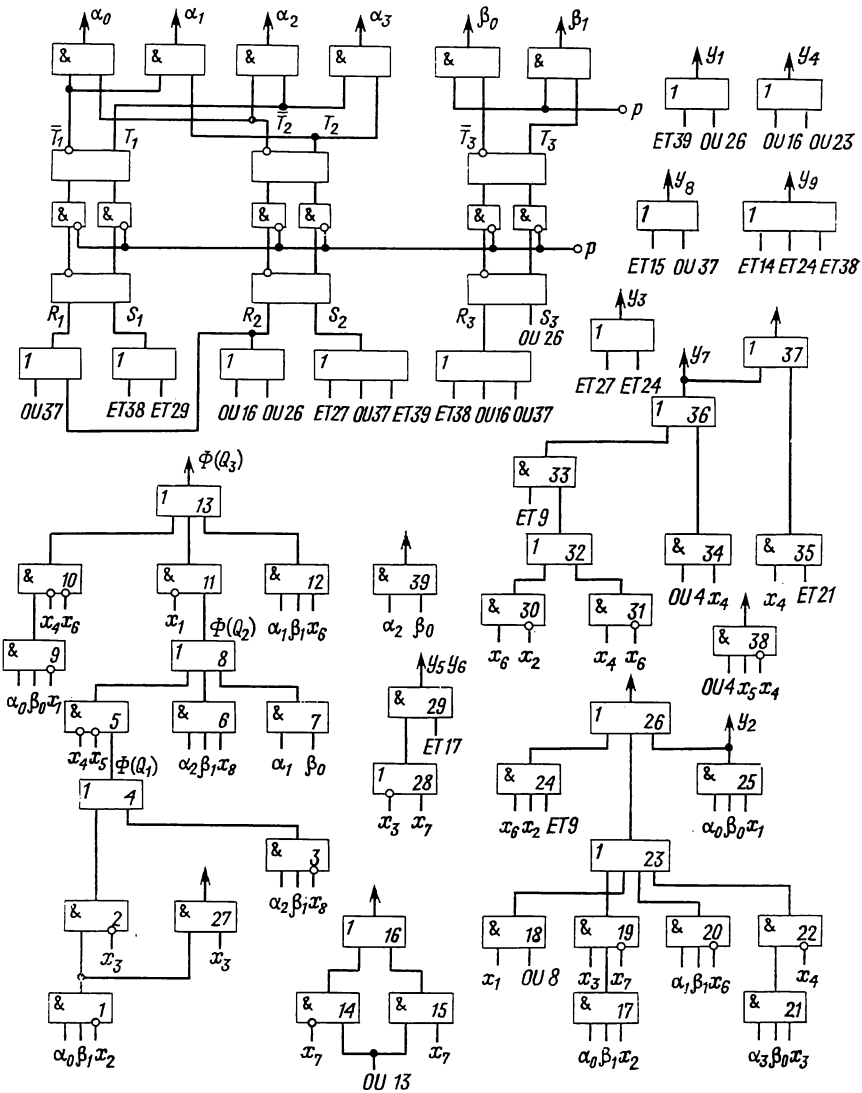


Fig. 9-21. Schéma logique de l'automate de Mealy construit d'après le tableau 9-4

tion $\widetilde{F}_{mt}$. Remarquons que cela devient possible seulement après complétion des fonctions d'excitation dans le n° 8 : en effet, avant le

n° 8, les circuits ET réunis sur le circuit OU_{mt} y fournissaient des fonctions d'excitation différentes (fig. 9-20, *b*).

10. Si $\bigcap_{t=1}^{T} Y_{mt} \neq \varnothing$, prévoir un circuit OU_m et lui appliquer les sorties de tous les circuits OU_{mt} ($t = 1, \ldots, T$). Le circuit OU_m devra élaborer les signaux des micro-opérations comprises dans $Y_m = \bigcap_{t=1}^{T} Y_{mt}$ et la fonction d'excitation $\widetilde{F}_m = \bigcup_{t=1}^{T} \widetilde{F}_{mt}$ complétée sur toutes les transitions vers a_m. Les circuits OU_{mt} ne produiront désormais que les signaux des micro-opérations contenues dans $Y_{mt} \setminus Y_m$, sans élaborer la fonction $\widetilde{F}_{mt}$, car nous avons complété les fonctions d'excitation sur toutes les transitions aboutissant à a_m (fig. 9-20, *c*).

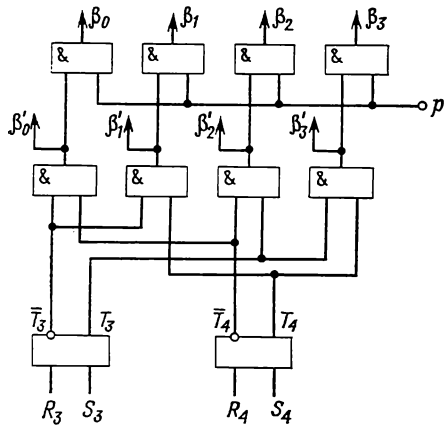


Fig. 9-22. Prédécodeur proposé pour la synthèse de l'automate de Moore

Au lieu des pas n°s 6 à 10, il serait préférable de faire, pour les signaux de sortie, une factorisation vers le bas combinée à une complétion des fonctions d'excitation comme décrit en fin du § 9-5; or, les opérations que cette procédure implique ne sont pas faciles à faire à la main.

11. Après avoir effectué les pas n°s 2 à 10 pour les $N + M$ ensembles de chemins conduisant vers les nœuds et les états, faire si possible une factorisation vers le bas pour tous les circuits ET impliqués.

12. Prévoir un circuit OU pour chaque signal de micro-opération et chaque composante des fonctions d'excitation qui est produit plus d'une fois dans le schéma réalisé.

13. Faire si possible une factorisation vers le bas pour tous les circuits OU du n° 12.

Le schéma logique de l'automate de Mealy construit d'après le tableau 9-4 est donné sur la figure 9-21. Afin d'éviter les courses et d'assurer la stabilité des états, on a réalisé un schéma à mémoire double.

Disons quelques mots sur la synthèse du schéma logique d'un automate microprogrammé de Moore d'après sa table de structure. La réservation des circuits pour les chemins de transition vers les nœuds se fait comme décrit dans les n°s 1 à 4 pour l'automate de Mealy. Les fonctions d'excitation sont réalisées et complétées s'il

y a lieu en analogie complète avec le cas de l'automate de Mealy. Quant aux signaux de sortie, on sait que dans le modèle de Moore les signaux des micro-opérations intervenant dans une micro-instruction donnée sont produits tant que l'automate se trouve dans l'état correspondant ; pour chaque état associé à une micro-instruction non vide, on construit donc une conjonction qui prend la valeur 1 quand l'automate se trouve dans l'état donné. Dans ce cas le schéma de la partie droite du prédécodeur se présente comme il est montré sur la figure 9-22. Ici le signal de sortie Y_t délivré par l'automate tant qu'il reste dans l'état a_s est égal à $\alpha^s\ (\beta^s)'$.

CHAPITRE 10

RÉALISATION MATRICIELLE DES AUTOMATES MICROPROGRAMMÉS

10-1. Réalisation matricielle des réseaux combinationnels

Les progrès réalisés ces dernières années par la technique des circuits intégrés ont permis d'atteindre un niveau d'intégration inouï, de sorte qu'il est devenu possible de mettre au point ce qu'on appelle l'intégration à grand échelle (IGE). Un cristal de tels circuits peut renfermer jusqu'à 10^5 composants. Vu la grande complexité de tels systèmes, il n'est plus possible d'envisager leur réalisation (et on peut affirmer que de nos jours la conception de circuits électriques et celle de systèmes sont inséparables) sans un emploi poussé, à toutes les étapes de la conception, de différents modèles mathématiques formels et d'ordinateurs.

A l'étape de la conception logique de systèmes numériques, se sont les unités de commande qui sont les plus difficiles à concevoir par suite de leur structure irrégulière et d'une grande diversité de blocs constitutifs. Sous ce rapport, sont particulièrement intéressantes les tentatives de régulariser la structure d'unités de commande sans pour autant introduire de redondance excessive. La réalisation matricielle de circuits logiques et d'automates est un pas important dans cette voie.

Considérons la réalisation matricielle de réseaux combinationnels sur l'exemple de la synthèse d'un système de fonctions booléennes données dans la forme canonique disjonctive (FCD):

$$\left.\begin{aligned} y_1 &= \bar{x}_1 x_2 \vee x_1 \bar{x}_2 \bar{x}_4 \vee x_1 x_3 x_4 ; \\ y_2 &= \bar{x}_1 x_2 \vee \bar{x}_1 \bar{x}_3 x_4 \vee \bar{x}_1 \bar{x}_2 x_3 \vee x_2 \bar{x}_4 ; \\ y_3 &= x_1 \bar{x}_2 \bar{x}_4 \vee x_1 \bar{x}_3 \vee \bar{x}_1 \bar{x}_2 x_3 . \end{aligned}\right\} \qquad (10\text{-}1)$$

Toute FCD d'un système de fonctions booléennes $y_1, \ldots, y_N$ des variables $x_1, \ldots, x_L$ peut être réalisée par un réseau à deux niveaux : au premier niveau sont formés tous les termes $e_1, \ldots, e_H$, et au deuxième, les disjonctions $y_1, \ldots, y_N$ de ces termes. Pour le système de fonctions (10-1), le réseau en question comprendra au premier niveau les sept termes $e_1, \ldots, e_7$, et au deuxième, les trois disjonctions y_1, y_2, y_3 des termes $e_1, \ldots, e_7$.

Avec l'IGE, les réseaux combinationnels sont souvent réalisés sous forme matricielle (fig. 10-1). On voit que M_1 et M_2 sont deux quadrilages ; aux intersections des barres on peut placer un élément à semi-conducteur : un transistor ou une diode. Chaque barre horizontale de la matrice M_1 est associée à un terme e_h du système de fonctions, et chaque barre verticale de la matrice M_2, à une disjonction y_n des termes obtenus dans la matrice M_1. On peut dire que la matrice M_1 forme H conjonctions différentes, à partir desquelles la matrice M_2 réalise N disjonctions différentes. Les matrices M_1 et M_2 sont construites avec des éléments à semi-conducteurs bipolaires ou des éléments M-D-SC *).

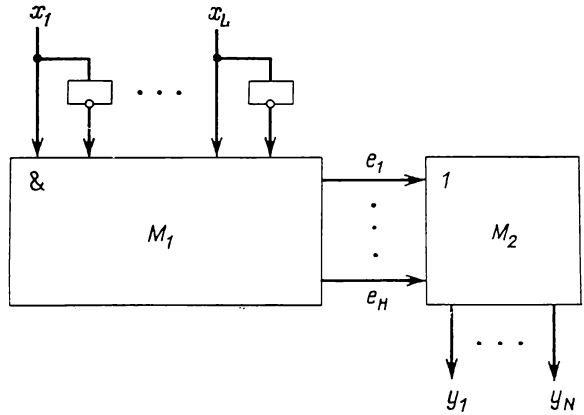


Fig. 10-1. Schéma synoptique de la réalisation matricielle des circuits combinationnels

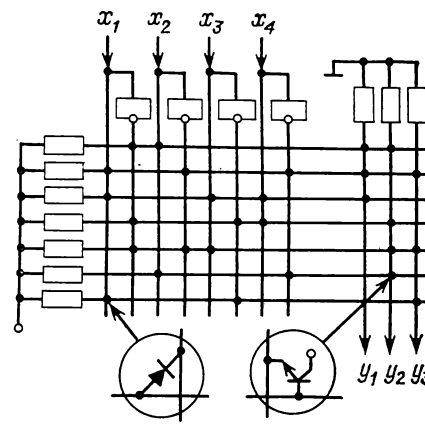


Fig. 10-2. Réalisation matricielle des expressions (10-1) avec des éléments bipolaires

Les figures 10-2 et 10-3 montrent chacune une réalisation matricielle des expressions (10-1) avec respectivement des éléments bipolaires et des éléments M-D-SC. Notons que sur la figure 10-3 chaque

*) La notation M-D-SC est utilisée pour métal-diélectrique-semi-conducteur. Le fonctionnement des éléments M-D-SC est basé sur le transfert des seuls porteurs des charges majoritaires, contrairement au cas des éléments bipolaires.

barre horizontale de la matrice M_1 et chaque barre verticale de la matrice M_2 réalisent la fonction logique « négation d'une disjonction » (OU-NON) *), fonction qui est fonctionnellement complète en ce sens que toute fonction booléenne peut être représentée à l'aide de la seule fonction OU-NON. Ainsi, par exemple, une conjonction peut

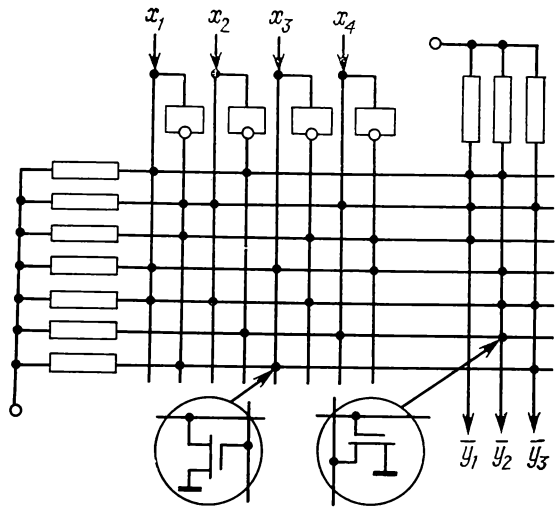


Fig. 10-3. Réalisation matricielle des expressions (10-2) avec des éléments M-D-SC

être représentée comme une fonction OU-NON des valeurs inverses des variables d'entrée. Exemple : $e_2 = x_1\bar{x}_2\bar{x}_4 = \overline{\bar{x}_1 \vee x_2 \vee x_4}$. De cette façon, chaque barre horizontale de la matrice M_1 de la figure 10-3 est associée à un terme de l'expression (10-1). Un transistor M-D-SC est placé à l'intersection de l'horizontale e_h et de la verticale x_l si la variable x_l apparaît dans le terme e_h sous forme complémentée et à l'intersection de l'horizontale e_h et de la verticale $\bar{x}_l$ si x_l apparaît dans e_h sous forme directe. Pour obtenir les fonctions y_n en code droit, on doit inverser les sorties de la matrice M_2, car chacune des disjonctions est réalisée comme la négation de la fonction OU-NON appropriée. Par la suite, pour ne pas compliquer l'exposé nous allons toujours utiliser les éléments bipolaires des matrices comme indiqué sur la figure 10-2. L'exemple ci-dessus suffit, à notre avis, pour que le lecteur puisse se faire une idée nette de la réalisation d'une matrice dans la base OU-NON.

*) Elle porte également le nom de fonction NI ou de fonction de Peirce.— *N. de t.*

Il est d'usage d'évaluer la complexité d'une réalisation matricielle de réseaux combinationnels par la capacité informationnelle globale (la surface) des matrices, laquelle pour la structure de la figure 10-1 est égale à

$$S(M) = S(M_1) + S(M_2) = 2LH + HN \text{ (bits)}.$$

On cherche à réduire la capacité d'information. A cet effet, le système de fonctions qu'on se propose de réaliser doit être représenté dans la forme canonique disjonctive comportant un nombre minimal de termes.

Il est d'usage de représenter les matrices M_1 et M_2 pour un système de fonctions booléennes donné par une table dont les colonnes sont les variables $x_1, \ldots, x_L$ et les fonctions $y_1, \ldots, y_N$. A chaque terme e_h on associe une ligne de la table. A l'intersection d'une colonne x_l ($l = 1, \ldots, L$) et d'une ligne e_h ($h = 1, \ldots, H$) on inscrit un « 1 » si la variable x_l apparaît dans le terme e_h sous sa forme directe, un « 0 » si x_l y apparaît sous sa forme complémentée ($\bar{x}_l$) et on met un tiret si la variable x_l n'y figure pas. A l'intersection de la ligne e_h avec la colonne y_n ($n = 1, \ldots, N$) on inscrit un « 1» si le terme e_h est présent dans la fonction y_n et un point dans le cas contraire. Les matrices de la figure 10-3 sont représentées par le tableau 10-1. Cette forme de représentation de matrices est très utile lorsqu'on envisage de confier la tâche de réalisation matricielle de réseaux à un ordinateur.

Tableau 10-1

Représentation tabulaire du schéma de la figure 10-2

x_1	x_2	x_3	x_4	y_1	y_2	y_3
0	1	—	—	1	1	•
1	0	—	0	1	•	1
1	—	1	1	1	•	•
0	—	0	1	•	1	•
0	0	1	—	•	1	1
—	1	—	0	•	1	•
1	—	0	—	•	•	1

Tableau 10-2

Deux fonctions de trois variables

$x_1x_2x_3$	y_1y_2
000	00
001	10
010	10
011	01
100	10
101	01
110	01
111	11

Montrons qu'un système de fonctions booléennes est susceptible d'une réalisation matricielle non seulement lorsqu'il est représenté dans la forme canonique disjonctive, mais aussi sous forme parenthéti-

sée de profondeur d'emboîtement quelconque. On s'assure sans peine que les fonctions y_1 et y_2 données par le tableau 10-2 se laissent écrire sous la forme:

$$\left.\begin{aligned} y_1 &= \overline{(x_1x_2 \vee \bar{x}_1\bar{x}_2)}\,\bar{x}_3 \vee (x_1x_2 \vee \bar{x}_1\bar{x}_2)\,x_3\,; \\ y_2 &= x_1x_2 \vee \overline{(x_1x_2 \vee \bar{x}_1\bar{x}_2)}\,x_3. \end{aligned}\right\} \qquad (10\text{-}2)$$

Une réalisation matricielle des fonctions y_1 et y_2 directement d'après les expressions (10-2) est schématisée sur la figure 10-4,

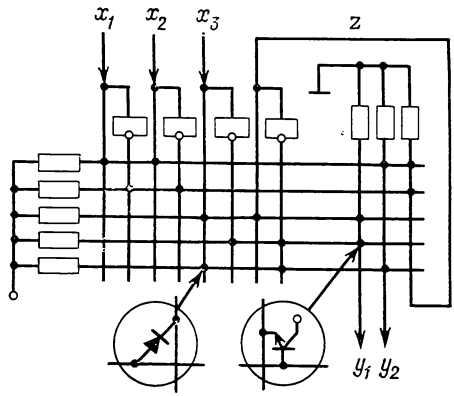


Fig. 10-4. Réalisation matricielle des formes parenthétisées

$z = x_1x_2 \vee \bar{x}_1\bar{x}_2$ y est la sortie de la matrice M_2 appliquée sur l'entrée de la matrice M_1.

10-2. Réalisation matricielle triviale d'un automate microprogrammé

Nous allons illustrer la réalisation matricielle d'un AMP sur l'exemple de synthèse de l'automate donné par le tableau 10-3 *). Rappelons que le tableau de ce type est le graphe des transitions de l'automate cherché, donné sous la forme d'une liste, chaque ligne du tableau correspondant à une transition.

Ainsi par exemple, en examinant la 8-ième ligne du tableau 10-3 on remarque qu'il existe dans l'automate une transition de l'état a_3 (code 1110) à l'état a_8 (code 0101) sous l'action de toute séquence d'entrée pour laquelle la fonction $\bar{x}_4\bar{x}_{10}$ vaut « 1 », avec apparition de « 1 » sur les sorties y_5, y_8, y_9 et de « 0 » sur toutes les autres sorties. Comme pour la transition en question les bascules T_2 et T_4 doivent

*) Les éléments de mémoire sont ici des bascules du type D.

Tableau 10-3

Table de structure de l'AMP

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t		D_h	h
a_1	0000	a_1	0000	$x_1\bar{x}_2$	y_1y_2	Y_1	—	1
		a_5	0001	$\bar{x}_1$	$y_1y_2y_{12}$	Y_2	D_4	2
		a_6	1010	$x_1x_2\bar{x}_3$	y_2y_{12}	Y_7	D_1D_3	3
		a_3	1110	$x_1x_2x_3$	y_1y_2	Y_1	$D_1D_2D_3$	4
a_2	0111	a_2	0111	x_6	—	Y_0	$D_2D_3D_4$	5
		a_3	1110	$\bar{x}_6$	y_3y_5	Y_4	$D_1D_2D_3$	6
a_3	1110	a_3	1110	x_{10}	y_3y_5	Y_4	$D_1D_2D_3$	7
		a_8	0101	$\bar{x}_4\bar{x}_{10}$	$y_5y_8y_9$	Y_6	D_2D_4	8
		a_9	0110	$x_4\bar{x}_{10}$	$y_{10}y_{15}$	Y_{10}	D_2D_3	9
a_4	1101	a_4	1101	$\bar{x}_7\bar{x}_9$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	10
		a_6	1010	x_7	y_{13}	Y_3	D_1D_3	11
		a_8	0101	$\bar{x}_7\bar{x}_9$	$y_{13}y_{14}$	Y_{13}	D_2D_4	12
a_5	0001	a_5	0001	$\bar{x}_1$	y_7y_{11}	Y_8	D_4	13
		a_6	1010	x_1	$y_{16}y_{17}$	Y_{12}	D_1D_3	14
a_6	1010	a_1	0000	x_5	y_1y_2	Y_1	—	15
		a_1	0000	$\bar{x}_5$	$y_{16}y_{17}$	Y_{12}	—	16
a_7	0010	a_2	0111	x_8	$y_{14}y_{18}$	Y_{15}	$D_2D_3D_4$	17
		a_4	1101	$\bar{x}_8$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	18
a_8	0101	a_4	1101	$\bar{x}_9$	y_6	Y_9	$D_1D_2D_4$	19
		a_7	0010	x_9	y_4y_6	Y_{11}	D_3	20
a_9	0110	a_2	0111	$\bar{x}_6x_{11}$	$y_5y_8y_9$	Y_6	$D_2D_3D_4$	21
		a_3	1110	$\bar{x}_{11}$	$y_3y_8y_9$	Y_{14}	$D_1D_2D_3$	22
		a_9	0110	x_6x_{11}	$y_{10}y_{15}$	Y_{10}	D_2D_3	23

basculer dans l'état « 1 » et y rester, il faut qu'elles reçoivent en entrée les signaux valant « 1 » (on trouvera les symboles D_2 et D_4 dans la colonne D_h).

On montre facilement qu'un automate microprogrammé peut être réalisé de façon triviale sous forme de la structure schématisée sur la figure 10-5. La matrice M_E (on a omis de représenter les inverseurs à ses entrées) forme les termes $e_1, \ldots, e_H$ correspondant aux diverses

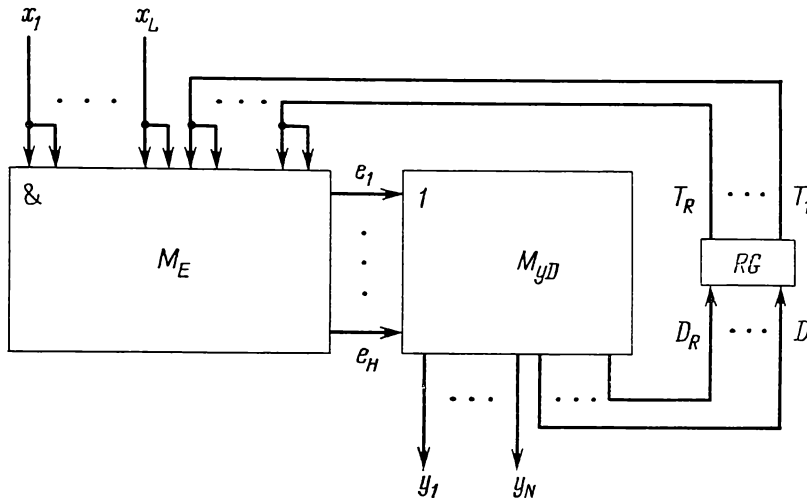


Fig. 10-5. Schéma synoptique d'une réalisation triviale des AMP

transitions (lignes de la table de structure) qui ont lieu dans l'automate, tandis que la matrice M_{yD} élabore les micro-opérations et les fonctions d'excitation des éléments de mémoire sous forme de disjonctions des termes appropriés choisis parmi les $e_1, \ldots, e_H$. Par exemple, si dans la ligne h de la table de structure on trouve la transition de l'état a_m à l'état a_s sous l'action du signal d'entrée X_h, il correspond à cette transition, dans la matrice M_E, la ligne réalisant le terme $e_h = A_m X_h$, où A_m est comme auparavant la conjonction correspondant au code $K(a_m)$ de l'état a_m. Ainsi, par exemple, on tire du tableau 10-3, $e_1 = \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4 x_1 x_2$, $e_2 = \overline{T}_1\overline{T}_2\overline{T}_3\overline{T}_4 x_1$, ..., $e_{23} = \overline{T}_1 T_2 T_3 \overline{T}_4 x_6 x_{11}$.

On conçoit que si une micro-opération y_n est élaborée au cours des transitions $n_1, \ldots, n_U$ de l'AMP (y_n est inscrite dans U lignes de la table de structure), dans la matrice M_{yD} lui est associée une barre verticale réalisant la disjonction

$$y_n = e_{n1} \vee \ldots \vee e_{nU} = \bigcup_{u=1}^{U} e_{nu}.$$

Les fonctions d'excitation des éléments de mémoire $D_1, \ldots, D_R$ sont réalisées d'une façon analogue, c'est-à-dire si une fonction D_r apparaît à la suite des transitions $r_1, \ldots, r_Q$ de l'automate (D_r figure dans Q lignes de sa table de structure), il lui correspond dans la matrice M_{yD} une verticale réalisant la disjonction

$$D_r = e_{r1} \vee \ldots \vee e_{rQ} = \bigvee_{q=1}^{Q} e_{rq}.$$

Ainsi, par exemple, dans le tableau 10-3 la microopération y_1 est inscrite dans les lignes 1, 2, 4 et 15, et la fonction D_1, dans les lignes 3, 4, 6, 7, 10, 11, 14, 18, 19 et 22. On a

$$\left.\begin{aligned} y_1 &= e_1 \vee e_2 \vee e_4 \vee e_{15}; \\ D_1 &= e_3 \vee e_4 \vee e_6 \vee e_7 \vee e_{10} \vee e_{11} \vee e_{14} \vee e_{18} \vee e_{19} \vee e_{22}. \end{aligned}\right\} \quad (10.3)$$

La réalisation matricielle triviale de l'automate microprogrammé spécifié par le tableau 10-3 est schématisée sur la figure 10-6. Ici et dans la suite, afin d'alléger les clichés, on omet de représenter les éléments de charge ; quant aux inverseurs, ils sont figurés par des triangles. Les matrices de la figure 10-6, de même que celles des figures 10-2 et 10-4, sont réalisées avec des éléments bipolaires, par conséquent, la réalisation est effectuée dans la base ET-OU.

Les avantages de la réalisation matricielle triviale sont évidents. Tout d'abord, c'est la simplicité de conception de l'automate, le travail se ramenant en fait à l'application de la table de structure sur les matrices M_E et M_{yD}. Notons que les diverses variantes de codification des états de l'automate ne compliquent pas le processus de son réalisation, ni son schéma, dont la complexité sera évaluée comme auparavant par la capacité informationnelle (surface) des matrices. On a dans le cas considéré

$$S(M) = S(M_E^*) + S(M_{yD}) = (2L + 2R)H + (N + R)H \text{ (bits)}.$$

L'inconvénient de la réalisation triviale consiste dans une redondance excessive des matrices M_E et M_{yD}. Cela tient au fait qu'en général, chaque transition concrète de l'automate d'un état dans un autre n'est définie que par les valeurs d'un petit nombre de variables appartenant à l'ensemble des variables d'entrée ; en outre, les valeurs « 1 » ne sont pas prises par tous les signaux de sortie ni par toutes les fonctions d'excitation, mais par leur sous-ensemble assez restreint. Par conséquent, les matrices de la figure 10-6 sont faiblement remplies.

Aussi cherche-t-on à réduire la capacité informationnelle globale (surface) des matrices lors de la réalisation des automates microprogrammés. On y parvient par différents procédés. Dans ce qui suit nous allons considérer quelques méthodes d'optimisation de schémas d'automates.

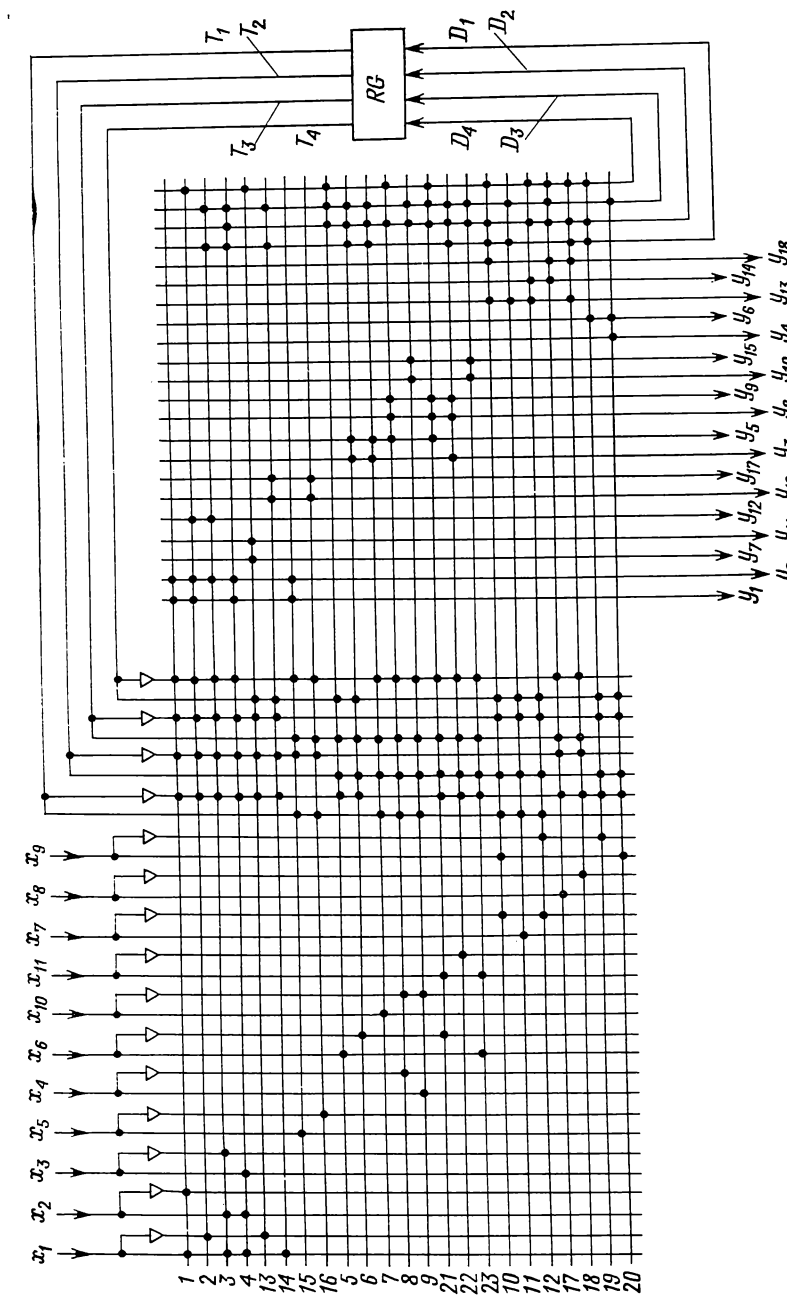


Fig. 10-6. Réalisation triviale des AMP

10-3. Passage à un nouvel ensemble de variables d'entrée lors d'une réalisation matricielle de l'AMP

Nous avons insisté plus haut sur le fait que le nombre de variables dont dépend essentiellement chaque transition dans l'AMP est faible devant l'ensemble des variables d'entrée $X = \{x_1, \ldots, x_L\}$. Si l'on ne tient pas compte de cette circonstance, on arrive à des matrices M_E trop encombrantes, comme celle de la figure 10-5. Une des méthodes d'optimisation de schémas d'automates est fondée sur le passage de l'ensemble des variables d'entrée X à un ensemble P, composé en général d'un nombre moindre de variables.

Désignons par $X(a_m)$ l'ensemble des variables d'entrée qui définissent toutes les transitions de l'automate à partir de l'état a_m. Dans notre exemple (voir tabl. 10-3):

$$\left.\begin{array}{ll} X(a_1)=\{x_1,\ x_2,\ x_3\}; & X(a_6)=\{x_5\}; \\ X(a_2)=\{x_6\}; & X(a_7)=\{x_8\}; \\ X(a_3)=\{x_4,\ x_{10}\}; & X(a_8)=\{x_9\}; \\ X(a_4)=\{x_7,\ x_9\}; & X(a_9)=\{x_6,\ x_{11}\}. \\ X(a_5)=\{x_1\}; & \end{array}\right\} \qquad (10\text{-}4)$$

On voit que le plus grand nombre de variables dont dépend une transition est égal à 3 (transitions à partir de l'état a_1). Désignons ce nombre dans le cas général par G et formons un nouvel ensemble de variables $P = \{p_1, \ldots, p_G\}$. Evidemment, $G \leqslant L$, mais en pratique on a le plus souvent G beaucoup plus petit que L. Dans notre exemple, $G = 3$, $L = 11$. On peut alors remplacer chaque variable de $X(a_m)$, $m = 1, \ldots, M$, par une variable de l'ensemble P de façon à avoir $p_g = x_l$ si dans l'état a_m la variable x_l a été remplacée par la variable p_g.

Supposons que, pour l'exemple considéré, on a effectué le passage à un nouvel ensemble de variables comme résumé dans le tableau 10-4. A l'intersection de la ligne a_m et de la colonne p_g on inscrit l'élément x_l si dans l'état a_m la variable x_l est remplacée par p_g. La ligne a_m contiendra tous les éléments de l'ensemble $X(a_m)$, chacun devant être mentionné une fois.

Du tableau 10-4 on voit, en particulier, que p_1 est égal à x_1 dans les états a_1 et a_5, à x_4 dans l'état a_3 et à x_6 dans les états a_2 et a_9. L'expression de p_1 s'écrira alors

$$p_1 = A_1x_1 \vee A_2x_6 \vee A_3x_4 \vee A_5x_1 \vee A_9x_6 \qquad (10\text{-}5)$$

En effet, si l'automate est dans l'état a_1, on a $A_2 = A_3 = A_5 = 0$, et $A_1 = 1$, donc $p_1 = x_1$. Les expressions du type (10-5) pour p_1, p_2 et p_3 se déduisent directement du tableau 10-4. Un tiret

à l'intersection d'une ligne a_m et d'une colonne p_g traduit le fait que la fonction p_g est incomplètement spécifiée, c'est-à-dire que dans l'état a_m elle peut être égale à toute variable de l'ensemble d'entrée. Ainsi, par exemple, en regard de l'état a_4 (voir tabl. 10-4), on a mis un tiret dans la colonne p_1 et on a inscrit les variables x_7 et x_9 dans les colonnes p_2 et p_3 respectivement, car toutes les transitions de l'automate de l'état a_4 sont définies par les seules variables x_7 et x_9. Après le changement de variables, ces transitions sont définies par les variables p_2 et p_3 et ne dépendent donc pas de la variable p_1 qui peut de ce fait prendre, pour l'état a_4, la valeur de n'importe laquelle des variables d'entrée. De même, la variable p_1 n'est pas spécifiée dans les états a_6, a_7 et a_8. Grâce à cette circonstance, il est possible d'ajouter à l'expression (10-5) les différents termes A_4x_t, A_6x_t, A_7x_t et A_8x_t en vue d'une simplification de p_1. Bien que la variable x_t appartienne à l'ensemble des variables d'entrée $X = \{x_1, \ldots, x_L\}$ de l'automate, il sera montré plus bas qu'il convient de choisir x_t parmi les seules variables de la colonne p_1. On a

Tableau 10-4

Changement de variables pour l'automate donné par le tableau 10-3

a_m \ p_g	p_1	p_2	p_3
a_1	x_1	x_2	x_3
a_2	x_6	—	—
a_3	x_4	x_{10}	—
a_4	—	x_7	x_9
a_5	x_1	—	—
a_6	—	x_5	—
a_7	—	—	x_8
a_8	—	—	x_9
a_9	x_6	—	x_{11}

$$p_1 = A_1x_1 \vee A_2x_6 \vee A_3x_4 \vee A_5x_1 \vee A_9x_6 \vee \\ \vee [(A_4 \vee A_6 \vee A_7 \vee A_8)(x_1 \vee x_4 \vee x_6)]. \qquad (10\text{-}6)$$

Le crochet met en relief les termes ajoutés en vue de compléter l'expression de p_1. Les expressions du type (10-6) pour p_1, p_2 et p_3 se laissent déduire directement du tableau 10-4:

$$\left.\begin{aligned}
p_1 &= (A_1 \vee A_5 \vee [A_4 \vee A_6 \vee A_7 \vee A_8])\, x_1 \vee (A_3 \vee [A_4 \vee A_6 \vee \\
&\quad \vee A_7 \vee A_8])\, x_4 \vee (A_2 \vee A_9 \vee [A_4 \vee A_6 \vee A_7 \vee A_8])\, x_6\,; \\
p_2 &= (A_1 \vee [A_2 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_2 \vee \\
&\quad \vee (A_6 \vee [A_2 \vee A_5 \vee A_7 \vee \\
&\quad \vee A_8 \vee A_9])\, x_5 \vee (A_4 \vee [A_2 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_7 \vee \\
&\quad \vee (A_3 \vee [A_2 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_{10}; \\
p_3 &= (A_1 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_3 \vee (A_7 \vee [A_2 \vee A_3 \vee A_5 \vee \\
&\quad \vee A_5 \vee A_6])\, x_8 \vee (A_4 \vee A_8 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_9 \vee \\
&\quad \vee (A_9 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_{11}.
\end{aligned}\right\} \qquad (10\text{-}7)$$

Sans nous arrêter sur les questions liées au codage des états d'automates microprogrammés, supposons que les états de l'automate choisi pour illustrer l'exposé sont codés comme indiqué dans

Tableau 10-5

Codage des états

T_1T_2 \ T_3T_4	00	01	11	10
00	a_1	a_5		a_7
01		a_8	a_2	a_9
11		a_4		a_3
10				a_6

le tableau 10-5. On a alors pour les variables p_1, p_2 et p_3 les expressions suivantes:

$$\left.\begin{aligned} p_1 &= \overline{T}_3x_1 \vee T_1x_4 \vee \overline{T}_1T_2x_6\,; \\ p_2 &= \overline{T}_2\overline{T}_3x_2 \vee T_1\overline{T}_2x_5 \vee T_4x_7 \vee T_2T_3x_{10}\,; \\ p_3 &= \overline{T}_2\overline{T}_3x_3 \vee \overline{T}_2T_3x_8 \vee T_4x_9 \vee T_2T_3x_{11}. \end{aligned}\right\} \qquad (10\text{-}8)$$

Les expressions (10-8) peuvent également être réalisées sous la forme de deux matrices, comme montré sur la figure 10-7. Si dans la table de structure de l'AMP (voir tabl. 10-3) on remplace les variables $x_1, \ldots, x_{11}$ par les variables p_1, p_2, p_3 conformément au tableau 10-4, on obtient le tableau 10-6. Dans la colonne P_h de ce dernier tableau on écrit le signal d'entrée apparaissant à la h-ième transition de l'automate et correspondant au signal d'entrée X_h du tableau 10-3 avant le changement de variables. Par exemple, à la transition (a_3, a_8) inscrite à la huitième ligne du tableau 10-3 correspond le signal d'entrée $X_8 = \overline{x}_4\overline{x}_{10}$. Comme dans l'état a_3 la variable x_4 est remplacée par la variable p_1 et la variable x_{10} par p_2 (voir tabl. 10-4), on trouve, à la huitième ligne du tableau 10-6, $P_8 = \overline{p}_1\overline{p}_2$.

Le schéma synoptique de l'AMP avec le changement de variables est montré sur la figure 10-8. Les matrices M_z et M_p réalisent les expressions du type (10-8), la matrice M_F, les termes $F_h = A_mP_h$, tandis que la matrice M_{yD} est la même que sur la figure 10-5. Remar-

Tableau 10-6

Table de structure de l'AMP après le changement de variables

a_m	$K(a_m)$	a_s	$K(a_s)$	P_h	Y_t		D_h	h
a_1	0000	a_1	0000	$\bar{p}_1\bar{p}_2$	y_1y_2	Y_1	—	1
		a_5	0001	$\bar{p}_1$	$y_1y_2y_{12}$	Y_2	D_4	2
		a_6	1010	$p_1p_2\bar{p}_3$	y_2y_{12}	Y_7	D_1D_3	3
		a_3	1110	$p_1p_2p_3$	y_1y_2	Y_1	$D_1D_2D_3$	4
a_2	0111	a_2	0111	p_1	—	Y_0	$D_2D_3D_4$	5
		a_3	1110	$\bar{p}_1$	y_3y_5	Y_4	$D_1D_2D_3$	6
a_3	1110	a_3	1110	p_2	y_3y_5	Y_4	$D_1D_2D_3$	7
		a_8	0101	$\bar{p}_1\bar{p}_2$	$y_5y_8y_9$	Y_6	D_2D_4	8
		a_9	0110	$p_1\bar{p}_2$	$y_{10}y_{15}$	Y_{10}	D_2D_3	9
a_4	1101	a_4	1101	$\bar{p}_2p_3$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	10
		a_6	1010	p_2	y_{13}	Y_3	D_1D_3	11
		a_8	0101	$\bar{p}_2\bar{p}_3$	$y_{13}y_{14}$	Y_{13}	D_2D_4	12
a_5	0001	a_5	0001	$\bar{p}_1$	y_7y_{11}	Y_8	D_4	13
		a_6	1010	p_1	$y_{16}y_{17}$	Y_{12}	D_1D_3	14
a_6	1010	a_1	0000	p_2	y_1y_2	Y_1	—	15
		a_1	0000	$\bar{p}_2$	$y_{16}y_{17}$	Y_{12}	—	16
a_7	0010	a_2	0111	p_3	$y_{14}y_{18}$	Y_{15}	$D_2D_3D_4$	17
		a_4	1101	$\bar{p}_3$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	18
a_8	0101	a_4	1101	$\bar{p}_3$	y_6	Y_9	$D_1D_2D_4$	19
		a_7	0010	p_3	y_4y_6	Y_{11}	D_3	20
a_9	0110	a_2	0111	$\bar{p}_1p_3$	$y_5y_8y_9$	Y_6	$D_2D_3D_4$	21
		a_3	1110	$\bar{p}_3$	$y_3y_8y_9$	Y_{14}	$D_1D_2D_3$	22
		a_9	0110	p_1p_3	$y_{10}y_{15}$	Y_{10}	D_2D_3	23

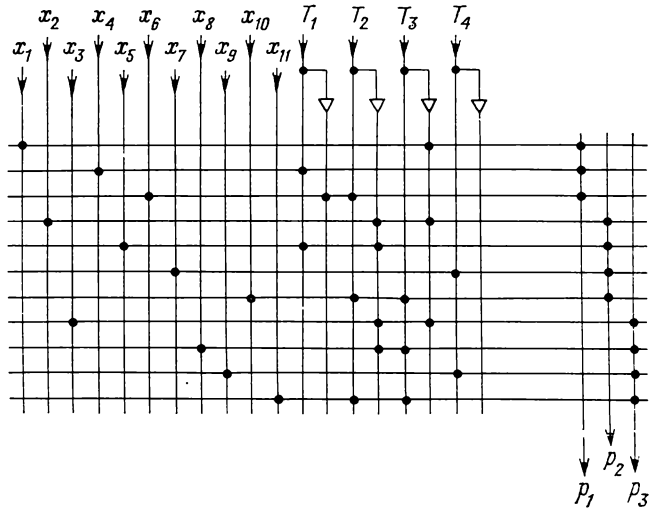


Fig. 10-7. Réalisation des fonctions p_1, p_2, p_3

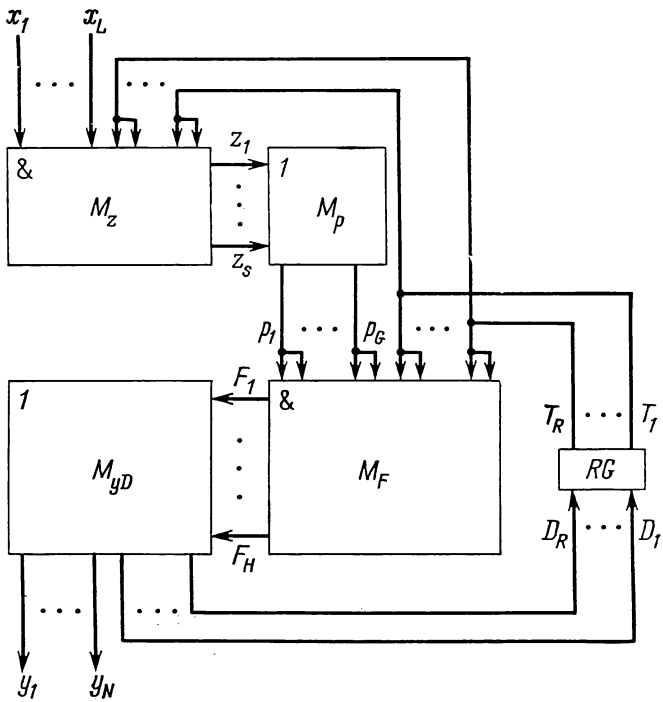


Fig. 10-8. Configuration d'un AMP après changement de variables

quons que dans la matrice M_z une seule barre verticale suffit pour représenter une variable d'entrée, car dans les expressions (10-7) ces variables ne sont pas complémentées.

Comparant les figures 10-5 et 10-8 on voit que la matrice M_E est remplacée par les matrices M_z, M_p et M_F. On gagne par le changement de variables d'entrée si la capacité d'information de la matrice M_E est plus grande que la capacité d'information globale des matrices M_z, M_p et M_F. Dans l'exemple considéré, $S(M_z) + S(M_p) + S(M_F) = (L + 2R)S + SG + (2G + 2R)H = 564$ bits et $S(M_E) = (2L + 2R)H = 690$ bits pour une réalisation triviale. Un gain modeste de 18 % est tout à fait naturel pour un automate aussi simple que celui que nous avons examiné à titre d'exemple. Dans les AMP réels de complexité moyenne, de paramètres $L = S = 50$, $H = 200$, $R = 6$ et $G = 5$, on a $S(M_E) = 22\,400$ bits, ce qui fait plus que doubler la quantité $S(M_z) + S(M_p) + S(M_F) = 7750$ bits.

Tableau 10-7

Autre variante de changement de variables

a_m \ p_g	p_1	p_2	p_3
a_1	x_1	x_5	x_3
a_2	x_6	—	—
a_3	x_4	x_{10}	—
a_4	x_9	—	x_7
a_5	x_1	—	—
a_6	—	x_5	—
a_7	—	—	x_8
a_8	—	—	x_9
a_9	x_{11}	—	x_6

Il existe plus d'une façon de faire le changement de variables d'entrée et de coder les états de l'automate; par conséquent, on obtiendra les différentes expressions du type (10-8) et donc les différentes valeurs de $S(M_z)$ et $S(M_p)$. Il importe donc de faire le changement de variables et de coder les états de l'AMP de la sorte que les surfaces des matrices M_z et M_p soient minimales possibles. Cette dernière condition est assurée par un nombre minimal de termes S dans les expressions du type (10-8). Notons que le nombre S ne dépasse pas celui des éléments (variables d'entrée) inscrits dans les cases du tableau du type 10-4. Autrement dit,

$$S \leqslant \sum_{m=1}^{M} |X(a_m)|.$$

Ici $|X(a_m)|$ est le nombre des éléments de l'ensemble $X(a_m)$, i.e. le nombre des variables d'entrée provoquant toutes les transitions possibles de l'automate à partir de l'état a_m.

Supposons que dans notre exemple les variables d'entrée sont remplacées comme montré dans le tableau 10-7. On remarque que les variables x_6 et x_9 sont inscrites aussi bien dans la colonne p_1

que dans la colonne p_3. On déduit directement du tableau 10-7 :

$$\left.\begin{aligned}
p_1 = {} & (A_1 \vee A_5 \vee [A_6 \vee A_7 \vee A_8])\, x_1 \vee \\
& \vee (A_3 \vee [A_6 \vee A_7 \vee A_8])\, x_4 \vee \\
& \vee (A_2 \vee [A_6 \vee A_7 \vee A_8])\, x_6 \vee \\
& \vee (A_4 \vee [A_6 \vee A_7 \vee A_8]) x_9 \vee \\
& \vee (A_9 \vee [A_6 \vee A_7 \vee A_8])\, x_{11}\,; \\
p_2 = {} & (A_1 \vee [A_2 \vee A_4 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_2 \vee \\
& \vee (A_6 \vee [A_2 \vee A_4 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_5 \vee \\
& \vee (A_3 \vee [A_2 \vee A_4 \vee A_5 \vee A_7 \vee A_8 \vee A_9])\, x_{10}; \\
p_3 = {} & (A_1 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_3 \vee \\
& \vee (A_9 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_6 \vee \\
& \vee (A_4 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_7 \vee \\
& \vee (A_7 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_8 \vee \\
& \vee (A_8 \vee [A_2 \vee A_3 \vee A_5 \vee A_6])\, x_9.
\end{aligned}\right\} \qquad (10\text{-}9)$$

Analysons les expressions (10-7) et (10-9). Supposons que dans les deux cas les états sont codés de la façon la meilleure en ce sens que toutes les expressions entre parenthèses fusionnent en une conjonction unique pour former exactement un terme. Chaque fonction p_g contiendra alors autant de termes qu'on trouve de variables différentes dans la colonne p_g de la table de changement de variables. Les tableaux 10-4 et 10-7 font voir que le système (10-7) doit comporter onze termes et le système (10-9), deux termes de plus, car les variables x_6 et x_9 sont présentes aussi bien dans la colonne p_1 que p_3.

La conclusion qu'on en tire est qu'il faut tâcher d'éviter qu'une variable x_l se rencontre dans plus d'une colonne p_g de la table de changement de variables, ce qui n'est d'ailleurs pas toujours possible.

De l'analyse des expressions (10-7) et (10-9) ressort clairement l'importance de coder les états de l'automate de façon à assurer que les disjonctions dans les parenthèses fusionnent en une conjonction unique. Dans notre exemple (voir tabl. 10-5), il a été possible de tenir compte de cette condition lors du codage des trois termes suivants des expressions (10-7) :

$$(A_1 \vee A_5 \vee [A_4 \vee A_6 \vee A_7 \vee A_8]);$$
$$(A_2 \vee A_9 \vee [A_4 \vee A_6 \vee A_7 \vee A_8]);$$
$$(A_4 \vee A_8 \vee [A_2 \vee A_3 \vee A_5 \vee A_6]).$$

Ainsi, la minimisation de la capacité d'information des matrices M_z et M_p dépend de la solution satisfaisante de deux problèmes :

1) recherche d'un changement de variables tel qu'il minimise le nombre total d'occurrences de chaque variable x_l dans les différentes colonnes p_g des tables du type 10-4 ;

2) choix d'un codage des états de l'automate tel qu'il assure le fusionnement le plus complet possible des conjonctions écrites dans les parenthèses des expressions du type (10-7) et donc minimise le nombre de termes différents dans le système de fonctions $p_1, \ldots, p_G$.

Ce sont là deux problèmes combinatoires complexes qu'on peut résoudre par examen complet des variantes. Pourtant, même pour un AMP de complexité moyenne, le volume des calculs est tellement grand que même un ordinateur ne peut en venir à bout en un temps raisonnable. En pratique, on résout ces problèmes par des méthodes approchées susceptibles de fournir des résultats voisins des optimaux.

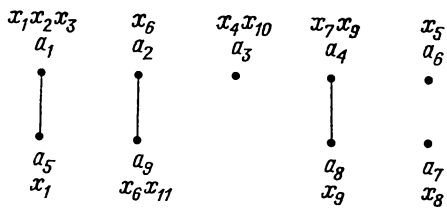


Fig. 10-9. Graphe de la relation ω

Montrons qu'il est possible de ramener le problème de changement de variables à plusieurs problèmes indépendants de dimension moindre. A cet effet, introduisons sur l'ensemble des états de l'AMP une relation ω que nous définissons comme suit : nous dirons que deux états a_i et a_j vérifient la relation ω si et seulement si les ensembles $X(a_i)$ et $X(a_j)$ ont des variables communes, i.e.

$$a_i \omega a_j \Longleftrightarrow X(a_i) \cap X(a_j) \neq \varnothing ; \quad i \neq j.$$

Construisons le graphe Γ_ω de la relation ω, en réunissant deux sommets a_i et a_j par une arête s'ils vérifient la relation ω (fig. 10-9). A côté de chaque sommet — état a_m — du graphe Γ_ω on a noté l'ensemble $X(a_m)$. Il ressort de la définition même de la relation ω que les différents sous-graphes ne peuvent comporter de variables d'entrée en commun ; aussi, en procédant à un changement de variables dans les états appartenant à un sous-graphe quelconque, n'a-t-on pas besoin de prendre en considération la façon dont seront remplacées les variables dans les états d'autres sous-graphes. Il est clair que le problème de changement de variables se décompose en autant de sous-problèmes plus simples qu'il existe de sous-graphes non connexes dans le graphe de la relation ω. La figure 10-9 met en évidence que, dans l'exemple considéré, il est possible de résoudre indépendamment l'un de l'autre six problèmes de changement de variables :

1) dans les états a_1 et a_5;
2) dans les états a_2 et a_9;
3) dans les états a_4 et a_8;
4) dans l'état a_3;
5) dans l'état a_6;
6) dans l'état a_7.

Il est naturel de s'attendre que la décomposition du problème de changement de variables ait pour effet de simplifier celui de codage des états de l'automate.

Considérons encore une variante de changement de variables dans l'exemple traité (tabl. 10-8). A la différence du tableau 10-4, le tableau 10-8 comporte dans la troisième colonne une variable unique x_3. On déduit directement du tableau 10-8:

$$\left.\begin{aligned}
p_1 = {} & (A_1 \vee A_5 \vee [A_6 \vee A_7 \vee A_8])\, x_1 \vee \\
& \vee (A_3 \vee [A_6 \vee A_7 \vee A_8])\, x_4 \vee \\
& \vee (A_2 \vee A_9 \vee [A_6 \vee A_7 \vee A_8])\, x_6 \vee \\
& \vee (A_4 \vee [A_6 \vee A_7 \vee A_8])\, x_7\,; \\
p_2 = {} & (A_1 \vee [A_2 \vee A_5])\, x_2 \vee \\
& \vee (A_6 \vee [A_2 \vee A_5])\, x_5 \vee \\
& \vee (A_7 \vee [A_2 \vee A_5]) x_8 \vee \\
& \vee (A_4 \vee A_8 \vee [A_2 \vee A_5])\, x_9 \vee \\
& \vee (A_3 \vee [A_2 \vee A_5])\, x_{10} \vee \\
& \vee (A_9 \vee [A_2 \vee A_5])\, x_{11}\,; \\
p_3 = {} & (A_1 \vee [A_2 \vee A_3 \vee A_4 \vee A_5 \vee A_6 \vee A_7 \vee \\
& \vee A_8 \vee A_9])\, x_3.
\end{aligned}\right\} \qquad (10\text{-}10)$$

En examinant les expressions (10-10) on remarque que $p_3 = x_3$, car $A_1 \vee A_2 \vee \ldots \vee A_9 = 1$. Il ne sert à rien d'introduire p_3: il vaut mieux d'utiliser à sa place la variable x_3 elle-même. Le tableau 10-8 se réduit alors au tableau 10-9. Dans la réalisation de l'AMP par la structure représentée sur la figure 10-8, la variable x_3 est introduite directement dans la matrice M_F. Avec le même codage des états de l'automate que celui employé auparavant (voir tabl. 10-5), on obtient:

$$\left.\begin{aligned}
p_1 &= \overline{T}_1\overline{T}_3 x_1 \vee T_1 T_3 x_4 \vee \overline{T}_1 T_3 x_6 \vee T_2\overline{T}_3 x_7\,; \\
p_2 &= \overline{T}_2\overline{T}_3 x_2 \vee T_1\overline{T}_2 x_5 \vee \overline{T}_1\overline{T}_2 T_3 x_8 \vee \\
&\quad \vee \overline{T}_3 T_4 x_9 \vee T_1 T_2 T_3 x_{10} \vee \overline{T}_1 T_2 T_3 x_{11}.
\end{aligned}\right\} \qquad (10\text{-}11)$$

Avec les expressions (10-11), la surface des matrices M_z et M_p sera visiblement moindre qu'avec les expressions (10-8).

Tableau 10-8

Encore une variante de changement de variables

a_m \ p_g	p_1	p_2	p_3
a_1	x_1	x_2	x_3
a_2	x_6	—	—
a_3	x_4	x_{10}	—
a_4	x_7	x_9	—
a_5	x_1	—	—
a_6	—	x_5	—
a_7	—	x_8	—
a_8	—	x_9	—
a_9	x_6	x_{11}	—

Tableau 10-9

Tableau 10-8 réduit

a_m \ p_g	p_1	p_2
a_1	x_1	x_2
a_2	x_6	—
a_3	x_4	x_{10}
a_4	x_7	x_9
a_5	x_1	—
a_6	—	x_5
a_7	—	x_8
a_8	—	x_9
a_9	x_6	x_{11}

Réduire le nombre de variables nouvellement introduites n'est possible que si le changement de variables permet de répartir les variables d'entrée dans le tableau du type 10-8 de telle façon que certaines colonnes en comportent une seule. Ces variables d'entrée seront alors appliquées directement sur l'entrée de la matrice M_F au lieu de variables p_g correspondantes. Par ailleurs, il faut toujours s'efforcer de minimiser le nombre d'occurrences de chacune des variables d'entrée dans les différentes colonnes du tableau du type 10-8. Ainsi, en effectuant le changement de variables pour l'exemple traité conformément au tableau 10-9, on écrira dans la colonne P_h du tableau 10-6, à la première, seconde, troisième et quatrième ligne respectivement, les signaux d'entrée

$$p_1\bar{p}_2, \quad \bar{p}_1, \quad p_1p_2\bar{x}_3, \quad p_1p_2x_3$$

sous l'effet desquels l'automate change d'état lorsqu'il a été placé dans l'état a_1.

10-4. Codage des micro-instructions

Rappelons que lors d'une réalisation triviale (voir fig. 10-5), la matrice M_E produit H termes $e_1, \ldots, e_H$ du type $e_h = A_mX_h$, définis par les transitions de l'AMP. Ici A_m est la conjonction correspondant au code de l'état initial a_m, X_h est le signal d'entrée inscrit à la h-ième ligne de la table de structure. La matrice M_{yD} réalise les micro-opérations y_n ($n = 1, \ldots, N$) et les fonctions d'excitation des éléments de mémoire D_r ($r = 1, \ldots, R$) comme

disjonctions des termes correspondant aux transitions de l'automate qui sont accompagnées de l'élaboration des fonctions y_n et D_r. Comme, en général, le nombre de micro-opérations associées à chaque transition concrète est de beaucoup inférieur au nombre total de micro-opérations de l'automate, la matrice M_{yD} se trouve être nécessairement trop redondante lors d'une réalisation matricielle triviale. Le codage des micro-instructions permet de remédier à cet inconvénient.

Codons les micro-instructions $Y_0, Y_1, \ldots, Y_T$ de l'automate en binaire: $K(Y_t) = (u_{t1} \ldots u_{tD})$, où D est la longueur du code donnée par l'expression $D =]\log_2(T+1)[$; u_{td} prend ses valeurs dans l'ensemble $\{0, 1\}$. Les micro-instructions qu'on rencontre dans l'exemple considéré (voir tabl. 10-3) sont les suivantes:

$$\left.\begin{array}{lll}
Y_0 = \varnothing; & Y_6 = \{y_5, y_8, y_9\}; & Y_{12} = \{y_{16}, y_{17}\}; \\
Y_1 = \{y_1, y_2\}; & Y_7 = \{y_2, y_{12}\}; & Y_{13} = \{y_{13}, y_{14}\}; \\
Y_2 = \{y_1, y_2, y_{12}\}; & Y_8 = \{y_7, y_{11}\}; & Y_{14} = \{y_3, y_8, y_9\}; \\
Y_3 = \{y_{13}\}; & Y_9 = \{y_6\}; & Y_{15} = \{y_{14}, y_{18}\}. \\
Y_4 = \{y_3, y_5\}; & Y_{10} = \{y_{10}, y_{15}\}; & \\
Y_5 = \{y_{13}, y_{18}\}; & Y_{11} = \{y_4, y_6\}; &
\end{array}\right\} \quad (10\text{-}12)$$

Désignons par $B_t = q_1^{u_{t1}} \ldots q_D^{u_{tD}}$ la conjonction correspondant au code $K(Y_t)$, $q_d^0 = \bar{q}_d$, $q_d^1 = q_d$, par analogie au codage des états, lorsque à chaque code de l'état a_m $(m = 1, \ldots, M)$ nous avons mis en correspondance une conjonction A_m. On a évidemment $B_t = 1$ uniquement dans le cas où c'est la micro-instruction Y_t qui se produit, tous les autres B_i $(i \neq t)$ étant à cette étape nuls.

Si une micro-opération y_n fait partie des micro-instructions $Y_\alpha, Y_\beta, \ldots, Y_\omega$, elle est définie par l'expression

$$y_n = B_\alpha \vee B_\beta \vee \ldots \vee B_\omega.$$

Comme, par exemple, $y_5 \in Y_4$ et $y_5 \in Y_6$, on a $y_5 = B_4 \vee B_6$. En d'autres mots,

$$y_n = c_{n1}B_1 \vee \ldots \vee c_{nT}B_T,$$

où $c_{nt} = 1$ si y_n appartient à la micro-instruction Y_t et $c_{nt} = 0$ dans le cas contraire.

Tableau 10-10

Apparition des micro-opérations de l'AMP dans les micro-instructions

y_n	$Y(y_n)$	y_n	$Y(y_n)$
y_1	Y_1, Y_2	y_{10}	Y_{10}
y_2	Y_1, Y_2, Y_7	y_{11}	Y_8
y_3	Y_4, Y_{14}	y_{12}	Y_2, Y_7
y_4	Y_{11}	y_{13}	Y_3, Y_5, Y_{13}
y_5	Y_4, Y_6	y_{14}	Y_{13}, Y_{15}
y_6	Y_9, Y_{11}	y_{15}	Y_{10}
y_7	Y_8	y_{16}	Y_{12}
y_8	Y_6, Y_{14}	y_{17}	Y_{12}
y_9	Y_6, Y_{14}	y_{18}	Y_5, Y_{15}

A chaque micro-opération y_n ($n=1, \ldots, 18$), le tableau 10-10 met en correspondance l'ensemble $Y(y_n)$ de micro-instructions dont y_n fait partie. On déduit directement de ce tableau:

$$\left.\begin{array}{lll}
y_1 = B_1 \vee B_2\,; & y_7 = B_8\,; & y_{13} = B_3 \vee B_5 \vee B_{13}\,; \\
y_2 = B_1 \vee B_2 \vee B_7\,; & y_8 = B_6 \vee B_{14}\,; & y_{14} = B_{13} \vee B_{15}\,; \\
y_3 = B_4 \vee B_{14}\,; & y_9 = B_6 \vee B_{14}\,; & y_{15} = B_{10}\,; \\
y_4 = B_{11}\,; & y_{10} = B_{10}\,; & y_{16} = B_{12}\,; \\
y_5 = B_4 \vee B_6\,; & y_{11} = B_8\,; & y_{17} = B_{12}\,; \\
y_6 = B_9 \vee B_{11}\,; & y_{12} = B_2 \vee B_7\,; & y_{18} = B_5 \vee B_{15}\,;
\end{array}\right\} \quad (10\text{-}13)$$

Une fois les micro-instructions codées, on compose le schéma matriciel de l'AMP sur la base de la structure représentée à la figure 10-10 où les matrices M_B et M_y réalisent les expressions du type (10-13). La matrice M_{qD} fournit les fonctions q_d ($d = 1, \ldots, D$) définies par l'expression

$$q_d = c_{d1}e_1 \vee \ldots \vee c_{dH}e_H,$$

où $e_h = A_m X_h$ est comme auparavant le terme correspondant à la h-ième transition; $c_{dh} = 1$ si le rang d du code de la micro-instruction, associée à la h-ième transition et donc inscrite à la h-ième ligne de la table de structure, vaut 1, et $c_{dh} = 0$ dans le cas contraire.

Ainsi, on réalise un gain avec le codage des micro-instructions si la capacité d'information de la matrice M_{yD} (voir fig. 10-7) est supérieure à la capacité d'information globale des matrices M_{qD},

M_B et M_y. Pour un automate de complexité moyenne, avec, par exemple, $H = 200$, $N = T = 50$ et $D = R = 6$, on a $S(M_{yD}) = H(N + R) = 11\,200$ bits et $S(M_{qD}) + S(M_B) + S(M_y) = H(D + R) + 2DT + TN = 5500$ bits, ce qui constitue moins que la moitié de $S(M_{yD})$.

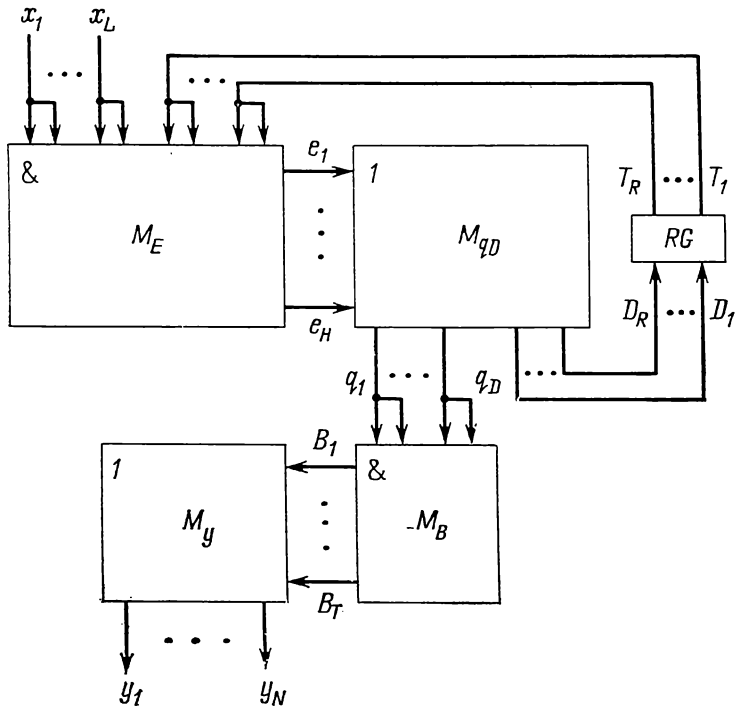


Fig. 10-10. Configuration d'un AMP après codage des micro-instructions

Voyons par quels procédés on peut réduire davantage la capacité d'information des matrices lorsqu'on fait appel au codage des micro-instructions.

1. En examinant les expressions (10-13) ou le tableau 10-10 on remarque que la micro-opération y_4, par exemple, appartient uniquement à Y_{11}, et donc $y_4 = B_{11}$, ce qui veut dire qu'on peut la recueillir directement sur la sortie de la matrice M_B. On peut procéder de même avec toutes les micro-opérations de l'ensemble

$$Y_B = \{y_7, y_{10}, y_{11}, y_{15}, y_{16}, y_{17}\},$$

qui se rencontrent chacune dans une seule micro-instruction. On parvient de cette façon à réduire la surface de la matrice M_y. On conçoit que si toutes les micro-opérations de l'automate s'expriment

à l'aide d'une conjonction unique, on n'a plus besoin de la matrice M_y, car elles sont formées toutes directement dans la matrice M_B. Il est maintenant clair que le but d'un codage de micro-instructions est de faire de la sorte que le plus grand nombre possible d'expressions de la forme (10-13) forment un terme unique. La solution exacte de ce problème est donnée dans [18]. Grâce au fait que les micro-opérations d'un AMP participent chacune dans un nombre relativement faible de micro-instructions, on arrive à supprimer, par un codage judicieux de celles-ci, la nécessité de la matrice M_y.

Notons que le codage des micro-instructions, tout comme le changement des variables, est un problème combinatoire complexe qu'on résout en pratique par des méthodes approchées euristiques, en général.

2. Cherchons toutes les micro-opérations (ou les ensembles de micro-opérations) qui sont inscrites à une seule ligne de la table de structure. Nous adressant au tableau 10-3, nous remarquons que la micro-opération y_4 y est écrite à la 20-ième ligne uniquement, ce qui signifie que $y_4 = e_{20}$ et donc on peut la prélever directement sur la 20-ième barre horizontale du montage de la figure 10-8.

Pour chercher l'ensemble Y_E de micro-opérations apparaissant chacune à une seule ligne de la table de structure de l'automate, on se servira utilement du tableau 10-11 qui à chaque micro-opération y_n ($n = 1, \ldots, N$) met en correspondance l'ensemble $E(y_n)$

Tableau 10-11

Apparition des termes $E(y_n)$ dans les micro-opérations y_n

y_n	$E(y_n)$	y_n	$E(y_n)$
y_1	e_1, e_2, e_4, e_{15}	y_{10}	e_9, e_{23}
y_2	$e_1, e_2, e_3, e_4, e_{15}$	y_{11}	e_{13}
y_3	e_6, e_7, e_{22}	y_{12}	e_2, e_3
y_4	e_{20}	y_{13}	$e_{10}, e_{11}, e_{12}, e_{18}$
y_5	e_6, e_7, e_8, e_{21}	y_{14}	e_{12}, e_{17}
y_6	e_{19}, e_{20}	y_{15}	e_9, e_{23}
y_7	e_{13}	y_{16}	e_{14}, e_{16}
y_8	e_8, e_{21}, e_{22}	y_{17}	e_{14}, e_{16}
y_9	e_8, e_{21}, e_{22}	y_{18}	e_{10}, e_{17}, e_{18}

des lignes du tableau 10-3, ou, en d'autres mots, l'ensemble des termes composant y_n. On en déduit immédiatement que l'ensemble en question est $Y_E = \{y_4, y_7, y_{11}\}$. Les micro-opérations de l'ensemble Y_E peuvent être prélevées directement sur la sortie de la matrice M_E de la figure 10-10.

3. Si une micro-instruction Y_t appartient à Y_E, i.e. $Y_t \subseteq Y_E$, et donc toutes les micro-opérations qui la composent sont disponibles directement sur les sorties de M_E, on n'a pas besoin de coder la micro-instruction en question (plus exactement, on lui associe un code composé de zéros, tout comme à Y_0). Il peut en résulter une réduction du nombre de positions (digits) D requises pour le codage de la micro-instruction (le nombre de barres des matrices M_q et M_B, fig. 10-10). Dans notre exemple n'est pas codée la micro-instruction $Y_8 \subseteq Y_E$.

Supposons que les micro-instructions soient codées comme l'indique le diagramme de Karnaugh du tableau 10-12.

Tableau 10-12

Codage des micro-instructions

q_1q_2 \ q_3q_4	00	01	11	10
00	Y_0	Y_2	Y_1	Y_{13}
01	Y_6	Y_7		Y_{14}
11	Y_4	Y_{10}	Y_3	Y_5
10	Y_{12}	Y_9	Y_{11}	Y_{15}

4. S'il existe des micro-opérations, non appartenant à Y_E, qui se rencontrent toujours ensemble dans différentes micro-instructions, on peut les prélever soit sur une même barre horizontale de M_B, soit sur une même barre verticale de M_y, d'où réduction de la surface des matrices. Le tableau 10-10 permet de reconnaître immédiatement de telles associations de micro-opérations. En effet, y_i et y_j se rencontrent toujours ensemble si $Y(y_i) = Y(y_j)$. Dans notre exemple, de tels cortèges sont $\{y_8, y_9\}$, $\{y_{10}, y_{15}\}$, $\{y_{16}, y_{17}\}$.

Pour un exemple aussi simple que le notre, on peut suivre le résultat de juxtaposition de variables dans les expressions du type (10-13) directement sur le diagramme de Karnaugh (tabl. 10-12).

$$\left.\begin{aligned}
y_1 &= B_1 \vee B_2 = \bar{q}_1\bar{q}_2q_4\,;\\
y_2 &= B_1 \vee B_2 \vee B_7 = \bar{q}_1q_4\,;\\
y_3 &= B_4 \vee B_{14} = q_1q_2\bar{q}_3\bar{q}_4 \vee \bar{q}_1q_2q_3\bar{q}_4\,;\\
y_5 &= B_4 \vee B_6 = q_2\bar{q}_3\bar{q}_4;\\
y_6 &= B_9 \vee B_{11} = q_1\bar{q}_2q_4;\\
y_8 &= y_9 = B_6 \vee B_{14} = \bar{q}_1q_2\bar{q}_4\,;\\
y_{10} &= y_{15} = B_{10} = q_1q_2\bar{q}_3q_4\,;\\
y_{12} &= B_2 \vee B_7 = \bar{q}_1\bar{q}_3q_4\,;\\
y_{13} &= B_3 \vee B_5 \vee B_{13} = q_1q_2q_3 \vee \bar{q}_1\bar{q}_2q_3\bar{q}_4\,;\\
y_{14} &= B_{13} \vee B_{15} = \bar{q}_2q_3\bar{q}_4;\\
y_{16} &= y_{17} = B_{12} = q_1\bar{q}_2\bar{q}_3\bar{q}_4\,;\\
y_{18} &= B_{15} \vee B_5 = q_1q_3\bar{q}_4.
\end{aligned}\right\} \qquad (10\text{-}14)$$

10-5. Construction du schéma logique

Après avoir effectué un changement de variables et un codage de micro-instructions, nous allons établir une nouvelle table de

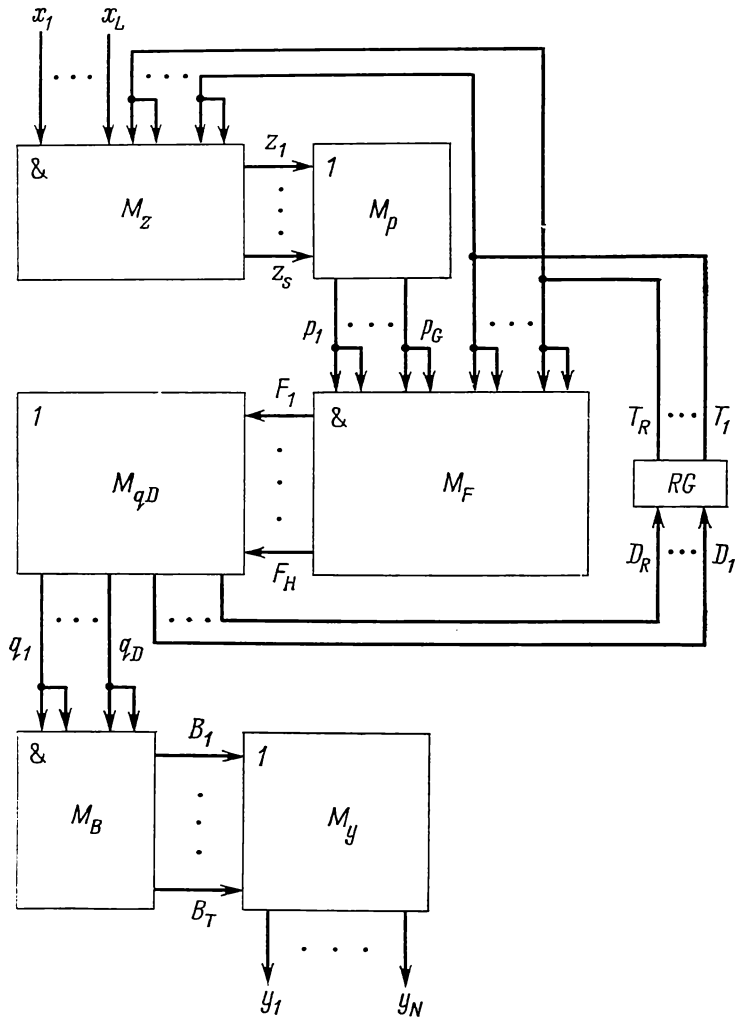


Fig. 10-11. Configuration d'un AMP après changement de variables et codage des micro-instructions

structure — le tableau 10-13 pour l'exemple traité. La colonne P_h de cette nouvelle table contient les signaux d'entrée, compte tenu du changement de variables conformément au tableau 10-9; en

outre, deux colonnes sont ajoutées : celle des codes de micro-instructions K (Y_t) du tableau 10-12 et une colonne des symboles de celles des composantes q_d qui prennent une valeur unité dans le code de la micro-instruction Y_t.

La figure 10-11 montre le schéma matriciel de l'AMP après le changement de variables et le codage de micro-instructions, la figure 10-12 représente la réalisation matricielle de notre modèle de l'automate, en conformité du tableau 10-13.

Les matrices M_z et M_p sont construites d'après les expressions (10-11). Les matrices M_F et M_{qD} comportent chacune 23 barres

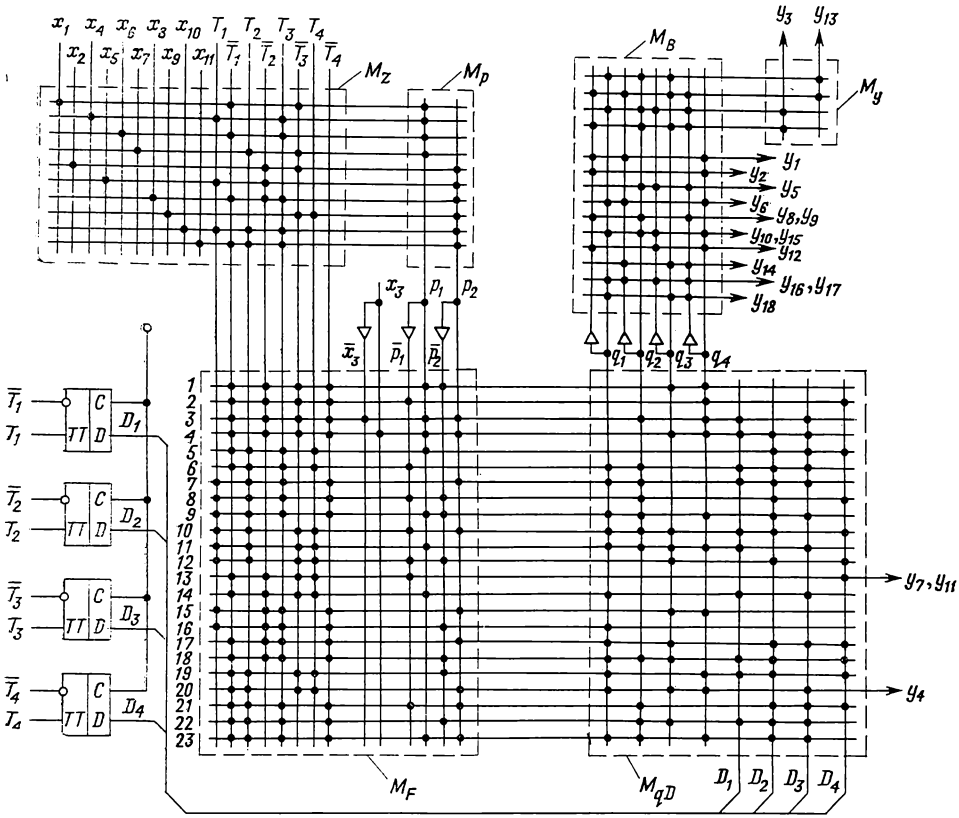


Fig. 10-12. Réalisation matricielle d'un AMP d'après le schéma de la figure 10-11

horizontales, chaque barre correspondant à une ligne de la table de structure (tabl. 10-13). Ici comme précédemment (voir fig. 10-2 et 10-4), les matrices M_z, M_F et M_B réalisant les conjonctions sont assemblées avec des diodes, et les matrices M_p et M_{qD}, pour les

Tableau 10-13

Table de structure de l'AMP après le changement de variables et le codage de micro-instructions

a_m	$K(a_m)$	a_s	$K(a_s)$	P_h	Y_t		D_h	$K(Y_t)$	q_d	h
a_1	0000	a_1	0000	$p_1\bar{p}_2$	y_1y_2	Y_1	—	0011	q_3q_4	1
		a_5	0001	$\bar{p}_1$	$y_1y_2y_{12}$	Y_2	D_4	0001	q_4	2
		a_6	1010	$p_1p_2\bar{x}_3$	y_2y_{12}	Y_7	D_1D_3	0101	q_2q_4	3
		a_3	1110	$p_1p_2x_3$	y_1y_2	Y_1	$D_1D_2D_3$	0011	q_3q_4	4
a_2	0111	a_2	0111	p_1	—	Y_0	$D_2D_3D_4$	0000	—	5
		a_3	1110	$\bar{p}_1$	y_3y_5	Y_4	$D_1D_2D_3$	1100	q_1q_2	6
a_3	1110	a_3	1110	p_2	y_3y_5	Y_4	$D_1D_2D_3$	1100	q_1q_2	7
		a_8	0101	$\bar{p}_1\bar{p}_2$	$y_5y_8y_9$	Y_6	D_2D_4	0100	q_2	8
		a_9	0110	$p_1\bar{p}_2$	$y_{10}y_{15}$	Y_{10}	D_2D_3	1101	$q_1q_2q_4$	9
a_4	1101	a_4	1101	$\bar{p}_1p_2$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	1110	$q_1q_2q_3$	10
		a_6	1010	p_1	y_{13}	Y_3	D_1D_3	1111	$q_1q_2q_3q_4$	11
		a_8	0101	$\bar{p}_1\bar{p}_2$	$y_{13}y_{14}$	Y_{13}	D_2D_4	0010	q_3	12
a_5	0001	a_5	0001	$\bar{p}_1$	y_7y_{11}	Y_8	D_4	—	—	13
		a_6	1010	p_1	$y_{16}y_{17}$	Y_{12}	D_1D_3	1000	q_1	14
a_6	1010	a_1	0000	p_2	y_1y_2	Y_1	—	0011	q_3q_4	15
		a_1	0000	$\bar{p}_2$	$y_{16}y_{17}$	Y_{12}	—	1000	q_1	16
a_7	0010	a_2	0111	p_2	$y_{14}y_{18}$	Y_{15}	$D_2D_3D_4$	1010	q_1q_3	17
		a_4	1101	$\bar{p}_2$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	1110	$q_1q_2q_3$	18
a_8	0101	a_4	1101	$\bar{p}_2$	y_6	Y_9	$D_1D_2D_4$	1001	q_1q_4	19
		a_7	0010	p_2	y_4y_6	Y_{11}	D_3	1011	$q_1q_3q_4$	20
a_9	0110	a_2	0111	$\bar{p}_1p_2$	$y_5y_8y_9$	Y_6	$D_2D_3D_4$	0100	q_2	21
		a_3	1110	$\bar{p}_2$	$y_3y_8y_9$	Y_{14}	$D_1D_2D_3$	0110	q_2q_3	22
		a_9	0110	p_1p_2	$y_{10}y_{15}$	Y_{10}	D_2D_3	1101	$q_1q_2q_4$	23

disjonctions, le sont avec des transistors. La matrice M_{qD} a 8 barres verticales: quatre pour les fonctions D_1, D_2, D_3 et D_4 d'excitation des éléments de mémoire, et quatre pour recevoir le code à quatre positions des micro-instructions. La matrice M_F comporte les diodes sur la première barre horizontale à l'intersection avec les barres verticales $\overline{T}_1$, $\overline{T}_2$, $\overline{T}_3$, $\overline{T}_4$, p_1 et $\overline{p}_2$, car le signal d'entrée inscrit à la première ligne du tableau 10-13 est $p_1\overline{p}_2$, et le code de l'état initial $K(a_1) = 0000$. Sur la première barre horizontale de la matrice M_{qD} on voit les transistors à l'intersection avec les barres verticales q_3, q_4, en conformité avec la première ligne du tableau 10-13: le code de la micro-instruction Y_1 produite lors de la transition qu'on trouve dans la première ligne du tableau vaut 0011. La première barre horizontale de la matrice M_{qD} ne comporte pas de transistors sur les verticales $D_1, \ldots, D_4$, car le code de l'état de transition $K(a_1) = 0000$ et les fonctions d'excitation essentielles sont nulles. De même, à la deuxième horizontale, M_F a les transistors sur les verticales $\overline{p}_1$, $\overline{T}_1$, $\overline{T}_2$, $\overline{T}_3$, $\overline{T}_4$ (le signal d'entrée écrit à la seconde ligne de la table de structure étant $\overline{p}_1$ et le code de l'état initial $K(a_1) = 0000$); dans M_{qD} les transistors sont sur les verticales q_4 ($K(Y_2) = 0001$) et D_4 ($K(a_5) = 0001$), et ainsi de suite. Les matrices M_B et M_y sont construites d'après les expressions (10-14). La matrice M_y s'est réduite à deux verticales pour les micro-opérations y_3 et y_{13}, les seules non-réalisables à l'aide d'une conjonction unique. Les micro-opérations y_4, y_7 et y_{11}, apparaissant une fois chacune dans la table de structure (tableau 10-13), sont prélevées directement sur les matrices M_F et M_{qD}.

10-6. Décomposition des matrices en réalisation triviale

Revenons au schéma de réalisation matricielle triviale montré à la figure 10-6. On remarque que la partie gauche de la matrice M_E (verticales associées aux variables $x_1, \ldots, x_{11}$) comporte peu d'éléments (points). Il en est de même de la partie gauche de la matrice M_{yD} avec les verticales $y_1, \ldots, y_{18}$. Intervertissons les barres horizontales et verticales de la matrice M_E de façon à en grouper les éléments autour de sa diagonale. Ceci fait, on peut essayer de transposer les verticales de la matrice M_{yD} dans le but de la diagonaliser, elle aussi. Le schéma de réalisation matricielle triviale prendra alors l'aspect montré sur la figure 10-13. On voit que les matrices M_E et M_{yD} se décomposent en trois sous-matrices chacune, par suite leur surface globale s'en trouve notablement réduite (fig. 10-14).

Montrons qu'on peut en venir à l'aide de raisonnements formels. Désignons par $E = \{e_1, \ldots, e_H\}$ l'ensemble des lignes de la table de structure de l'AMP, et par $X(e_h)$ et $Y(e_h)$ l'ensemble des varia-

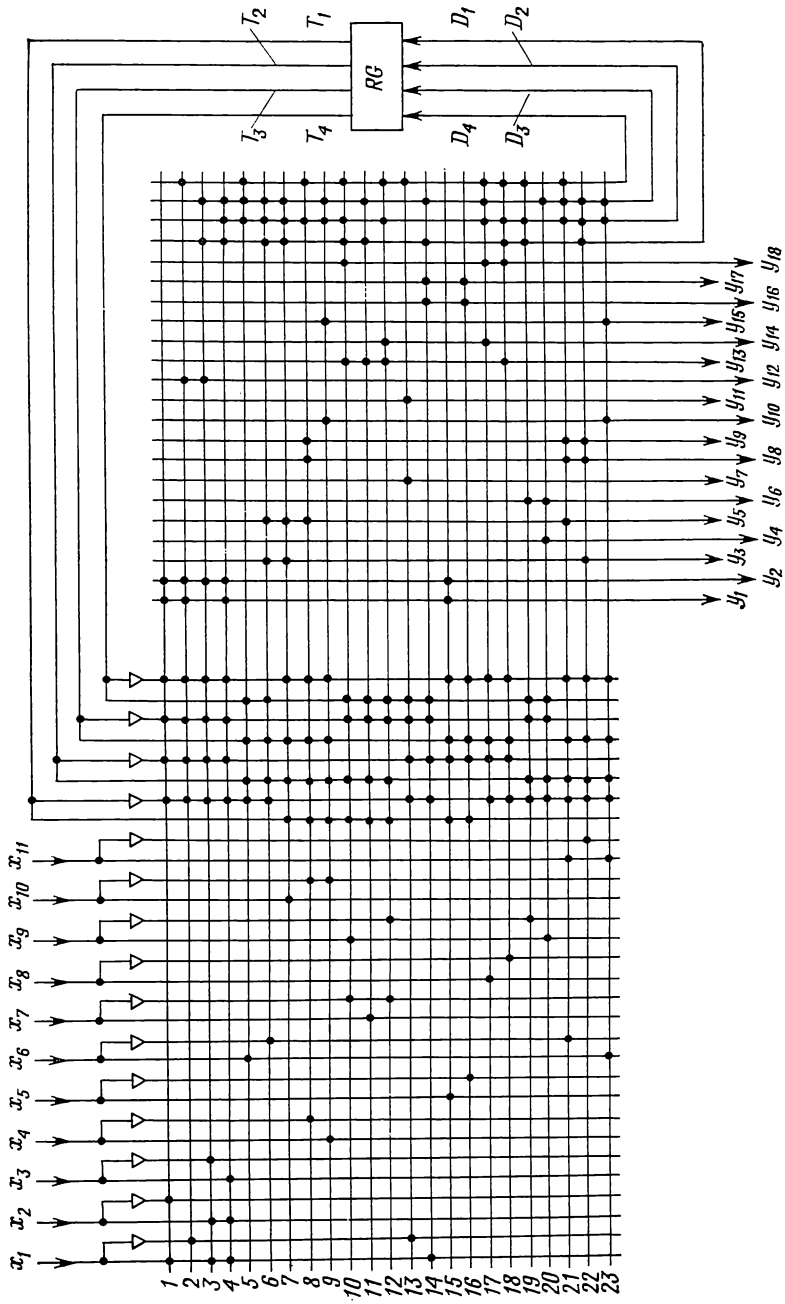


Fig. 10-13. Diagonalisation des matrices M_E et M_{yD}

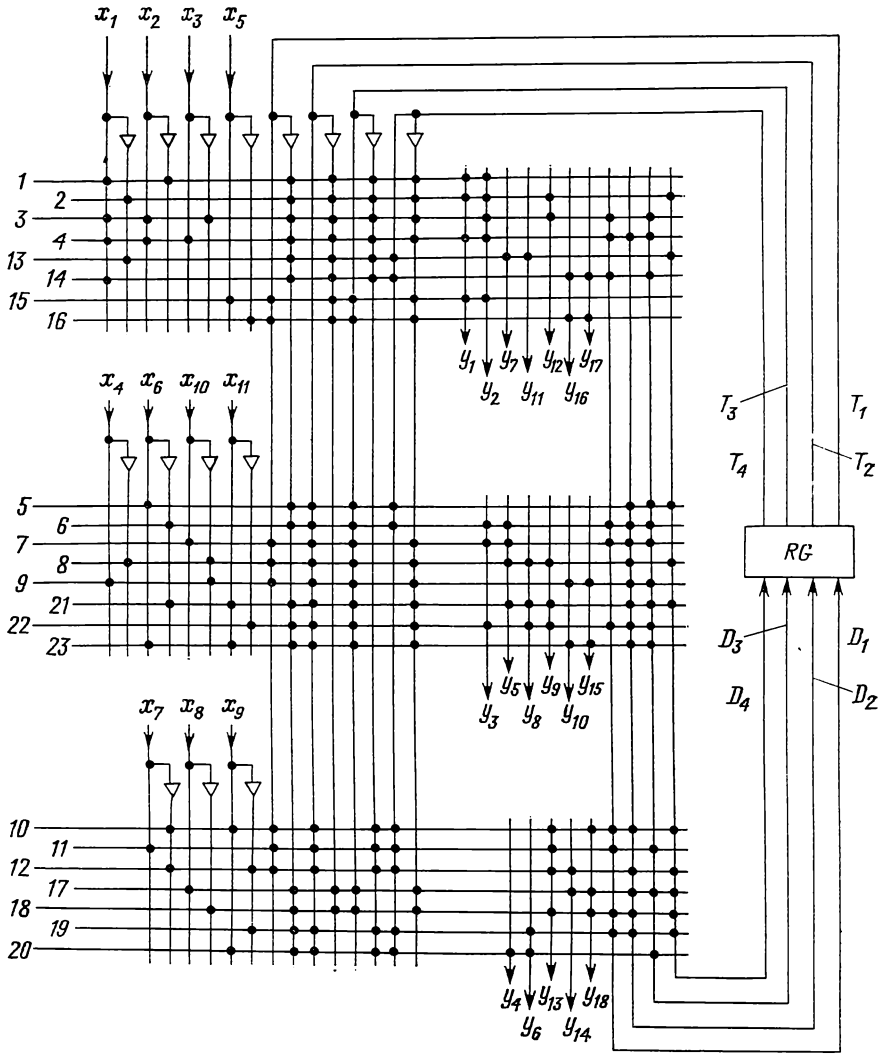


Fig. 10-14. Décomposition des matrices M_E et M_{yD}

bles d'entrée et celui des micro-opérations inscrits dans la h-ième ligne de cette table *). Introduisons maintenant sur l'ensemble E une relation γ que nous définissons comme suit: deux lignes e_i et e_j vérifient la relation γ si et seulement si l'intersection des ensembles $X(e_i)$ et $X(e_j)$ n'est pas vide, c'est-à-dire si au moins une variable d'entrée est inscrite à la fois dans la ligne e_i et dans la ligne e_j.

Le graphe Γ_γ de la relation γ pour l'AMP défini par le tableau 10-3 est montré sur la figure 10-15. La relation γ définit une partition

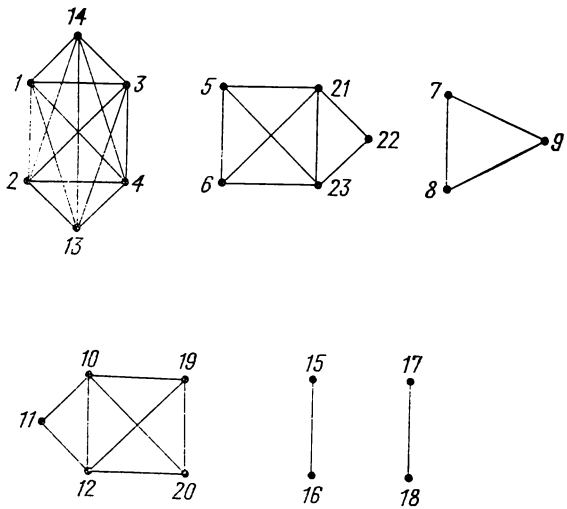


Fig. 10-15. Graphe de la relation γ

π_γ dont chaque ensemble comportera les sommets d'une même composante de connexité de Γ_γ. Dans notre exemple, $\pi_\gamma = \{E^1, E^2, E^3, E^4, E^5, E^6\} = \overline{1.2.3.4.13.14}, \overline{5.6.21.22.23}, \overline{7.8.9}, \overline{10.11.12.19.20}, \overline{15.16}, \overline{17.18}$. Désignons par $X(E^i)$ l'ensemble des variables d'entrée inscrites dans les lignes de la partie E^i. Il ressort des définitions de π_γ et de $X(E^i)$ que $X(E^i) \cap X(E^j) = \varnothing$ pour deux parties quelconques E^i et E^j $(i \neq j)$. On voit que la partition π_γ entraîne la partition $\pi_{\gamma x}$ de l'ensemble des variables d'entrée $X = \{x_1, \ldots, x_L\}$. Dans notre exemple $\pi_{\gamma x} = \overline{x_1 x_2 x_3}, \overline{x_4 x_{10}}, \overline{x_5},$

*) On a utilisé ici une même notation e_h pour désigner le terme correspondant à la h-ième ligne et la h-ième ligne elle-même. Cela ne prête pas à confusion, vu que les variables $X(e_h)$ inscrites dans la ligne e_h apparaissent toutes dans le terme e_h. La même remarque en ce qui concerne l'ensemble des variables de sortie $Y(e_h)$ (voir plus bas).

$\overline{x_6 x_{11}}$, $\overline{x_7 x_9}$, $\overline{x_8}$, ce qui veut dire que la matrice M_E de la réalisation triviale de l'AMP peut être décomposée en six sous-matrices disjointes, avec les verticales correspondant aux variables entrant dans chacune des parties de $\pi_{\gamma x}$.

D'une façon analogue on décompose la matrice M_{yD}. A cet effet introduisons sur l'ensemble E des lignes de la table de structure une relation τ telle que deux lignes e_i et e_j vérifient cette relation si et seulement si l'intersection des ensembles $Y(e_i)$ et $Y(e_j)$ n'est pas vide, c'est-à-dire si les lignes e_i et e_j comportent au moins une micro-opération en commun. Le graphe Γ_τ illustrant l'exemple con-

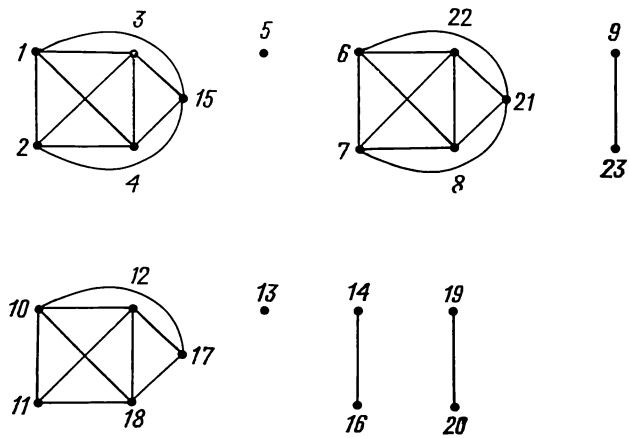


Fig. 10-16. Graphe de la relation τ

sidéré est donné sur la figure 10-16. On conçoit que si le graphe Γ_τ se décompose en sous-graphes non connexes, on peut définir une partition $\pi_{\tau y}$ dont chaque partie comportera les micro-opérations inscrites dans les lignes de la partie correspondante de la partition π_τ. Dans notre exemple $\pi_\tau = \overline{1.2.3.4.15}, \overline{5}, \overline{6.7.8.21.22}, \overline{9.23}, \overline{10.11.12.17.18}, \overline{13}, \overline{14.16}, \overline{19.20}$; $\pi_{\tau y} = \overline{y_1 y_2 y_{12}}, \overline{y_3 y_5 y_8 y_9}, \overline{y_4 y_6}, \overline{y_7 y_{11}}, \overline{y_{10} y_{15}}, \overline{y_{13} y_{14} y_{18}}, \overline{y_{16} y_{17}}$. On remarque que le nombre de parties de $\pi_{\tau y}$ est de un inférieur à celui de π_τ; ceci s'explique par le fait que l'ensemble des variables de sortie apparaissant dans la deuxième partie de π_τ (cinquième ligne du tableau 10-3) est vide. La matrice M_{yD} de notre exemple est donc décomposable en au moins sept sous-matrices, c'est-à-dire autant qu'il y a de parties dans la partition $\pi_{\tau y}$.

En réunissant les relations γ et τ, on peut obtenir une décomposition simultanée des matrices M_E et M_{yD}. A cet effet, introdui-

sons une relation β que nous définissons comme suit: deux lignes e_i et e_j de la table de structure vérifient cette relation si et seulement si $X(e_i) \cap X(e_j) \neq \varnothing$ ou $Y(e_i) \cap Y(e_j) \neq \varnothing$, c'est-à-dire que les lignes e_i et e_j comportent au moins une variable d'entrée ou au

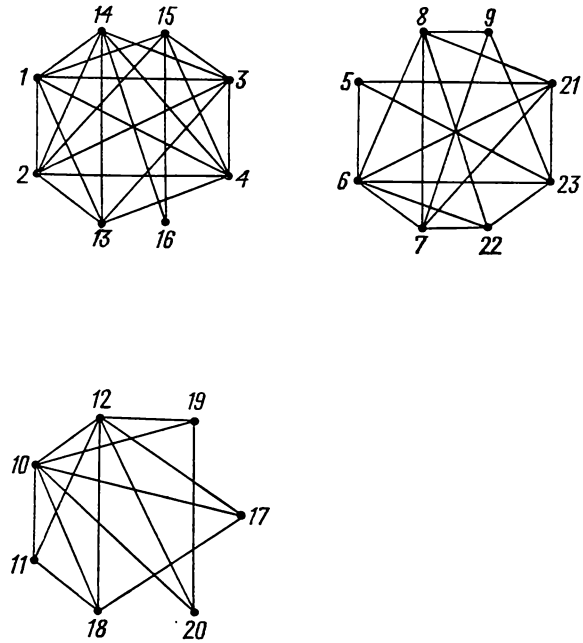


Fig. 10-17. Graphe de la relation β

moins une micro-opération en commun. Il est aisé de voir que $\beta = \gamma \cup \tau$ et $\pi_\beta = \pi_\gamma + \pi_\tau$. Le graphe Γ_β correspondant est donné sur la figure 10-17. Il se compose de trois sous-graphes définissant les ensembles des partitions π_β, $\pi_{\beta x}$ et $\pi_{\beta y}$:

$$\left.\begin{aligned}
\pi_\beta &= \overline{1.2.3.4.13.14.15.16},\ \overline{5.6.7.8.9.21.22.23},\\
&\qquad\qquad \overline{10.11.12.17.18.19.20};\\
\pi_{\beta x} &= \overline{x_1x_2x_3x_5},\ \overline{x_4x_6x_{10}x_{11}},\ \overline{x_7x_8x_9};\\
\pi_{\beta y} &= \overline{y_1y_2y_7y_{11}y_{12}y_{16}y_{17}},\ \overline{y_3y_5y_8y_9y_{10}y_{15}},\ \overline{y_4y_6y_{13}y_{14}y_{18}}.
\end{aligned}\right\} \qquad (10\text{-}15)$$

Les relations (10-15) font voir que la relation β permet de définir une partition de la table de structure de l'AMP telle que les variables d'entrée et celles de sortie se séparent (tabl. 10-14).

Tableau 10-1

Table de structure de l'AMP remaniée en fonction de la relation β

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t		D_h	h
a_1	0000	a_1	0000	$x_1\bar{x}_2$	y_1y_2	Y_1	—	1
		a_5	0001	$\bar{x}_1$	$y_1y_2y_{12}$	Y_2	D_4	2
		a_6	1010	$x_1x_2\bar{x}_3$	y_2y_{12}	Y_7	D_1D_3	3
		a_3	1110	$x_1x_2x_3$	y_1y_2	Y_1	$D_1D_2D_3$	4
a_5	0001	a_5	0001	$\bar{x}_1$	y_7y_{11}	Y_8	D_4	13
		a_6	1010	x_1	$y_{16}y_{17}$	Y_{12}	D_1D_3	14
a_6	1010	a_1	0000	x_5	y_1y_2	Y_1	—	15
		a_1	0000	$\bar{x}_5$	$y_{16}y_{17}$	Y_{12}	—	16
a_2	0111	a_2	0111	x_6	—	Y_0	$D_2D_3D_4$	5
		a_3	1110	$\bar{x}_6$	y_3y_5	Y_4	$D_1D_2D_3$	6
a_3	1110	a_3	1110	x_{10}	y_3y_5	Y_4	$D_1D_2D_3$	7
		a_8	0101	$\bar{x}_4\bar{x}_{10}$	$y_5y_8y_9$	Y_6	D_2D_4	8
		a_9	0110	$x_4\bar{x}_{10}$	$y_{10}y_{15}$	Y_{10}	D_2D_3	9
a_9	0110	a_2	0111	$\bar{x}_6x_{11}$	$y_5y_8y_9$	Y_6	$D_2D_3D_4$	21
		a_3	1110	$\bar{x}_{11}$	$y_3y_8y_9$	Y_{14}	$D_1D_2D_3$	22
		a_9	0110	x_6x_{11}	$y_{10}y_{15}$	Y_{10}	D_2D_3	23
a_4	1101	a_4	1101	$\bar{x}_7x_9$	$x_{13}y_{18}$	Y_5	$D_1D_2D_4$	10
		a_6	1010	x_7	y_{13}	Y_3	D_1D_3	11
		a_8	0101	$\bar{x}_7\bar{x}_9$	$y_{13}y_{14}$	Y_{13}	D_2D_4	12
a_7	0010	a_2	0111	x_8	$y_{14}y_{18}$	Y_{15}	$D_2D_3D_4$	17
		a_4	1101	$\bar{x}_8$	$y_{13}y_{18}$	Y_5	$D_1D_2D_4$	18
a_8	0101	a_4	1101	$\bar{x}_9$	y_6	Y_6	$D_1D_2D_4$	19
		a_7	0010	x_9	y_4y_6	Y_{11}	D_3	20

Pour décomposer les matrices M_E et M_{yD} on procède d'une façon analogue (voir fig. 10-14). Notons que le tableau 10-14 permet d'effectuer le changement de variables et le codage des micro-opérations indépendamment pour chacune de ses parties. Si le nombre d'ensembles de la partition π (ici π désigne n'importe laquelle des partitions dont il a été question ci-dessus) s'avère être trop élevé et s'il est désirable de décomposer les matrices en un nombre moindre de sous-matrices, il convient de réunir certains ensembles de π. On obtient de la sorte une partition $\pi' > \pi$ avec un nombre voulu d'ensembles dont on se servira pour la décomposition des matrices.

CHAPITRE 11

SYNTHÈSE DES AUTOMATES MICROPROGRAMMÉS AVEC DES MATRICES LOGIQUES PROGRAMMABLES

11-1. Matrices logiques programmables

Outre les circuits intégrés IGE à structure matricielle individualisée que nous avons considérés au chapitre précédent, l'industrie électronique moderne produit une gamme étendue de matrices logiques programmables (MLP). Une MLP représente un bloc fonctionnel, fabriqué selon une technique des circuits imprimés avec des diodes et transistors, destiné à être utilisé pour la réalisation de circuits logiques de systèmes numériques. Un intérêt accru ces derniers temps pour les MLP tient à plusieurs raisons [24]:

1) en raison d'une complexité toujours croissante du logiciel, la tendance est de réaliser de plus en plus de fonctions logiques des ordinateurs modernes sous forme de circuits;

2) la technologie des circuits intégrés a bénéficié ces derniers temps de nouvelles solutions logiques et techniques dont l'effet a été entre autres d'augmenter la capacité d'information des MLP;

3) sont implantées en pratique des méthodes de conception automatisée assistée par ordinateur;

4) avec la possibilité pour le client de programmer lui-même ses MLP, la voie est toute tracée pour les utiliser dans le matériel de destination diverse.

Selon leur organisation, on distingue les MLP combinationnelles et les MLP à mémoire.

Une MLP combinationnelle (fig. 11-1, *a*) se compose de deux matrices M_1 et M_2, la première délivrant q termes de s variables (complémentées ou non), et la seconde, t disjonctions des termes fournis par la matrice M_1. La MLP représentée sur la figure 11-1, *a* est un réseau combinationnel; aussi tout système de fonctions booléennes $y_1, \ldots, y_N$ ($N \leqslant t$) de variables $x_1, \ldots, x_L$ ($L \leqslant s$) dont la forme normale disjonctive comporte H termes ($H \leqslant q$) peut-il être réalisé à l'aide d'une seule MLP. Au cours de ce chapitre, les matrices logiques programmables combinationnelles seront désignées MLP (s, t, q) et figurées sur les dessins comme montré sur la figure 11-1, *b*.

A la différence d'une MLP (s, t, q), une MLP à mémoire (fig. 11-2, a) possède dans le circuit de retour, entre les matrices M_2 et M_1, un registre RG à r positions binaires. La matrice M_1

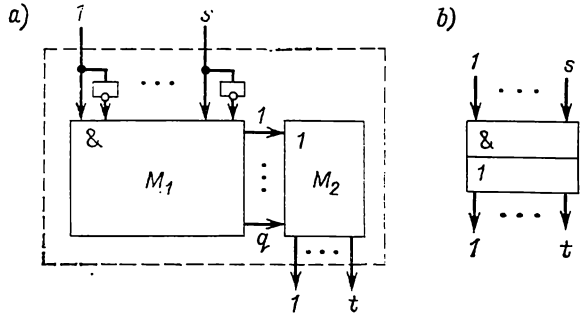


Fig. 11-1. Une MLP de logique combinationnelle (a) ; son symbole graphique (b)

peut fournir q termes de $s + r$ variables (complémentées ou non), et la matrice M_2, $t + r$ disjonctions des termes obtenus dans la matrice M_1. Le nombre r d'éléments de mémoire du registre est choisi de façon que l'on ait $2^r \geqslant q$. La MLP représentée sur la

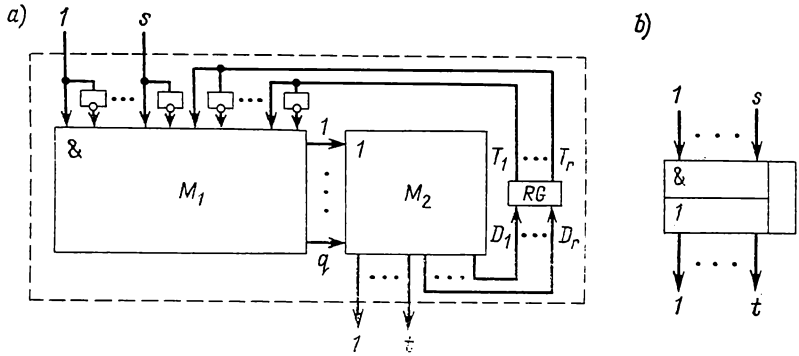


Fig. 11-2. Une MLP à mémoire (a) ; son symbole graphique (b)

figure 11-2, a est un circuit doté d'une mémoire ; une telle MLP suffit à elle seule à réaliser de façon triviale un automate à $L \leqslant s$ variables d'entrée, $N \leqslant t$ variables de sortie (micro-opérations) et $H \leqslant q$ transitions, comme dans le cas du montage de la figure 10-5. Une matrice logique programmée à mémoire sera désignée MLP (s, t, q, r) et figurée comme montré sur la figure 11-2, b.

L'industrie électronique soviétique produit actuellement en série les MLP des deux types avec $12 \leqslant s \simeq t \leqslant 24$, $48 \leqslant q \leqslant 96$, $r \leqslant 13$.

L'organisation d'une MLP combinationnelle ressemble à celle d'une mémoire à semi-conducteurs schématisée sur la figure 11-3. Les deux schémas diffèrent par le mode d'accès de la matrice M_2, qui est souple dans les MLP, ce qui facilite la réalisation des circuits logiques. Les avantages des MLP sont les suivants:

1) la matrice M_1 d'une MLP est programmable, à la différence du décodeur DC d'une mémoire à semi-conducteurs qui manque totalement de souplesse;

2) le nombre de sorties d'un décodeur est nécessairement maximal: $q = 2^s$. La matrice M_1 d'une MLP en possède moins: $q < 2^s$. Si dans la mémoire chaque séquence d'entrée provoque l'apparition d'un terme et d'un seul, i.e. sélectionne un mot et un seul dans l'ensemble des mots mémorisés, il en est tout autrement d'une MLP:

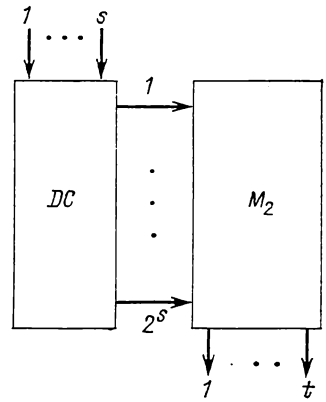


Fig. 11-3. Mémoire à semi-conducteurs

a) une partie de séquences d'entrée peut être non significative, i.e. n'exciter aucun terme, ne choisir aucun mot dans la matrice M_2, ce qui équivaut à la sélection dans la mémoire de mots composés de zéros;

b) chaque terme de MLP prend la valeur unité pour $k = 2^n$ séquences d'entrée, où n est le nombre d'entrées non essentielles pour le terme en question. Plusieurs séquences d'entrée peuvent donc exciter un même terme, i.e. choisir un même mot dans la matrice M_2;

c) une même séquence d'entrée peut provoquer l'excitation de plusieurs termes (leur faire prendre la valeur unité), ce qui fait que plusieurs mots sont sélectionnés à la fois dans la matrice M_2 et sur les barres de sortie on obtient leur disjonction rang par rang;

3) dans le cas d'une mémoire, l'adjonction d'une entrée a pour effet de doubler les dimensions de la matrice M_2; pour une MLP il n'en est pas de même: le nombre de ses entrées est limité par les possibilités de réalisation physique de la matrice M_1 et peut être assez grand, avec par ailleurs toujours $q \ll 2^s$.

En résumé, une mémoire réalise la table de vérité du système $y_1, \ldots, y_N$ de fonctions booléennes de l'ensemble $x_1, \ldots, x_L$ de variables d'entrée (N et L ne dépassant pas respectivement la longueur de mots et le nombre de positions de l'adresse mémoire). Quant à la MLP, elle réalise la forme normale disjonctive de ce système de fonctions, le nombre de termes distincts ne devant pas dépasser q, qui est le nombre de termes de la matrice M_1. D'où son champ d'application: la MLP est destinée à la réalisation de

fonctions à grand nombre de variables et à un nombre de termes relativement petit, i.e. de fonctions typiques de la majorité écrasante d'unités de commande réels.

Suivant le mode de programmation, qui est en fait le mode d'interconnexion de barres des matrices, on distingue deux types de MLP.

1. Les MLP programmées en cours de fabrication. Dans les MLP de ce type, les interconnexions des barres de matrices se font par masquage au dernier stade du processus de fabrication et ne peuvent pas être modifiées par la suite. Le plus souvent le masquage est utilisé dans la métallisation de certaines portions du cristal dans le but de réaliser des connexions des sorties des éléments aux barres des matrices de MLP. On dit parfois que c'est là une programmation par masquage.

2. Les MLP programmables par l'utilisateur. Les MLP de ce type sont livrées au client non programmées. Celui-ci procède à leur programmation au moyen de dispositifs spéciaux par suppression de connexions superflues (barrettes, jonctions de transistors, etc.). Ainsi, la fabrication des MLP se termine chez le client qui effectue également tous les contrôles et vérifications. Le procédé de programmation le plus souvent employé consiste à éliminer les interconnexions inutiles par brûlure des barrettes sous l'action d'un courant d'intensité élevée. Un autre procédé consiste à percer une jonction de transistors.

D'après les communications récentes, il existe à l'heure actuelle des MLP reprogrammables. La reprogrammation s'effectue par un procédé électrique après effacement du contenu de la MLP par un rayonnement ultra-violet (parfois aux rayons X) ou sous l'action d'un courant électrique, le procédé étant appliqué séparément pour chaque interconnexion concernée.

On peut étendre les possibilités fonctionnelles des MLP en réunissant dans un montage plusieurs MLP : on réunit soit les sorties, soit les barres assignées aux termes pour obtenir un schéma équivalent à une MLP à nombre élevé de sorties ou de termes respectivement.

Si l'on veut obtenir à partir d'une MLP (s, t, q) un montage qui possède kt sorties, il suffit de réunir les entrées de même nom de k MLP. Sur la figure 11-4 on donne à titre d'exemple un montage avec un nombre double de sorties qui est réalisé par l'assemblage de deux MLP (s, t, q). Le montage obtenu est équivalent à une MLP de paramètres s, $2t$ et q. En extension k-uple du nombre de sorties, les matrices M_1 de toutes les k MLP (s, t, q) sont identiques, tandis que les matrices M_2 permettent de réaliser un système de kt fonctions de s variables avec q termes.

Supposons que l'on se propose de réaliser, avec des MLP (10, 10, 20), un système de fonctions booléennes $y_1, \ldots, y_{20}$ de varia-

bles $x_1, \ldots, x_{10}$ dans la forme normale disjonctive (FND) qui comporte vingt termes distincts $e_1, \ldots, e_{20}$. Décomposons ce système d'une façon arbitraire en deux systèmes de dix fonctions chacun, disons $y_1, \ldots, y_{10}$ et $y_{11}, \ldots, y_{20}$. Les fonctions $y_1, \ldots, y_{10}$ dépendent d'au plus dix variables, tandis que leur FND comporte au plus 20 termes; par conséquent, les $y_1, \ldots, y_{10}$ peuvent être réalisées avec une seule MLP (10, 10, 20). Mais les fonctions $y_{11}, \ldots$ $\ldots, y_{20}$ peuvent elles aussi être réalisées avec une seule MLP (10,

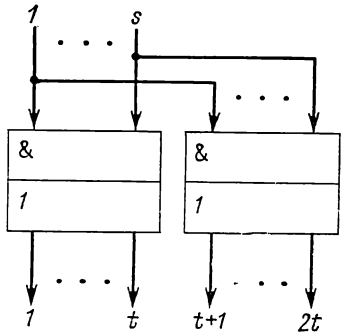


Fig. 11-4. Duplication du nombre de sorties d'une MLP

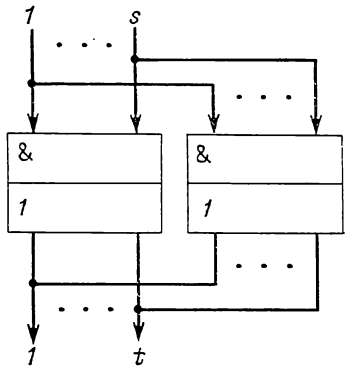


Fig. 11-5. Duplication du nombre de termes d'une MLP

10, 20). En réunissant donc deux MLP (10, 10, 20) comme montré sur la figure 11-4, on assure une réalisation triviale du système donné.

Si l'on cherche à obtenir une augmentation k-uple du nombre de termes avec des MLP (s, t, q), il convient de réunir aussi bien les entrées que les sorties de même nom de toutes les k MLP (s, t, q). Les sorties sont reliées entre elles par l'intermédiaire des circuits logiques OU. Dans le cas général, pour une augmentation k-uple du nombre de termes, on aura besoin de t circuits OU à k entrées. A l'heure actuelle sont commercialisées des MLP (s, t, q) dont les sorties peuvent être connectées entre elles directement. La figure 11-5 montre deux MLP (s, t, q) connectées en vue de réaliser une augmentation du nombre de termes. Le montage est équivalent à une MLP de paramètres s, t, $2q$. En extension k-uple du nombre de termes, les matrices M_1 de toutes les k MLP (s, t, q) réalisent kq termes distincts qu'on peut ensuite utiliser dans la FND de n'importe laquelle des t fonctions.

Soit à réaliser, avec des MLP (10, 10, 20), un système de fonctions booléennes $y_1, \ldots, y_{10}$ de variables $x_1, \ldots, x_{10}$. Il s'agit notamment d'obtenir la FND du système composée de 40 termes

distincts $e_1, \ldots, e_{40}$. Décomposons l'ensemble des termes $\{e_1, \ldots, e_{40}\}$ en deux sous-ensembles, disons $E_1 = \{e_1, \ldots, e_{20}\}$ et $E_2 = \{e_{21}, \ldots, e_{40}\}$. Obtenons E_1 avec l'une des MLP (10, 10, 20) et E_2 avec l'autre. Soit $y_1 = e_1 \vee e_2 \vee e_{21} \vee e_{22}$. On peut alors réaliser $y_1^1 = e_1 \vee e_2$ dans la première MLP (10, 10, 20) et $y_1^2 = e_{21} \vee e_{22}$ dans la seconde; la fonction y_1 se présente comme leur disjonction: $y_1 = y_1^1 \vee y_1^2$, on l'obtient en reliant les sorties y_1^1 et y_1^2 des MLP utilisées. Si l'on réunit deux MLP (10, 10, 20) comme montré sur la figure 11-5, on obtient une réalisation triviale du système de fonctions donné.

On peut procéder à une augmentation simultanée du nombre de termes et du nombre de sorties de MLP (s, t, q). Dans le cas d'une MLP à mémoire, lorsqu'il s'agit de k-upler le nombre de sorties, il convient de connecter, tout comme dans le cas de MLP (s, t, q), les entrées respectives de toutes les k MLP (s, t, q, r).

Réaliser avec des MLP (s, t, q) un système de fonctions booléennes de variables dont le nombre L dépasse le nombre s d'entrées de la MLP, ou un AMP avec $L > s$ variables d'entrée et $H > q$ transitions à l'aide des MLP (s, t, q, r) n'est pas aussi simple que cela peut paraître. Les réalisations des cas évoqués font appel à des méthodes de décomposition des fonctions booléennes et des automates microprogrammés.

11-2. Synthèse des automates microprogrammés avec des MLP combinationnelles

L'automate microprogrammé à réaliser sera comme auparavant spécifié à l'aide d'une table de structure du type du tableau 10-3. Soit H le nombre de lignes de la table de structure, R celui d'éléments de mémoire de l'automate et $X = \{x_1, \ldots, x_L\}$ et $Y = \{y_1, \ldots, y_N\}$ les ensembles de variables d'entrée et de sortie respectivement. On montre que si les paramètres de l'AMP vérifient les relations

$$L + R \leqslant s, \quad N + R \leqslant t, \quad H \leqslant q, \tag{11-1}$$

on peut en envisager la réalisation triviale (fig. 11-6) avec une seule MLP (s, t, q), directement d'après sa table de structure, tout comme dans le cas de la réalisation triviale matricielle. Les rôles des matrices M_1 et M_2 de la MLP (s, t, q) concernée sont répartis comme suit: M_1 fournit l'ensemble des termes $E = \{e_1, \ldots, e_H\}$ et M_2 l'ensemble $\{y_1, \ldots, y_N, D_1, \ldots, D_R\}$ des fonctions de sortie et des fonctions d'excitation de la mémoire de l'automate. Ainsi, si l'application du signal d'entrée X_h produit la transition $a_m \to a_s$ inscrite dans la h-ième ligne de la table de structure de l'automate, il correspond à cette transition le terme de la MLP (s, t, q) réalisant

la conjonction $e_h = A_m X_h$, où A_m est la conjonction correspondant au code $K(a_m)$ de l'état a_m. Si, dans la table de structure, la micro-opération y_n est inscrite dans les lignes de numéros $n_1, \ldots, n_U$ et la fonction d'excitation D_r dans les lignes de numéros $r_1, \ldots, r_Q$, alors

$$y_n = e_{n1}, \ldots, e_{nU};$$
$$D_r = e_{r1}, \ldots, e_{rQ}.$$

Lorsque les paramètres de l'automate ne vérifient pas une au moins des relations (11-1), on aura besoin pour le réaliser de plusieurs MLP (s, t, q) reliées entre elles de façon appropriée. Supposons pour fixer les idées que seule la seconde relation de (11-1) n'est pas vérifiée. On peut alors envisager une réalisation de l'automate basée sur l'extension du nombre de sorties de MLP (s, t, q), avec le coefficient d'extension

$$k_1 = \left]\frac{N+R}{t}\right[.$$

Dans le cas où c'est le nombre H de lignes de la table de structure qui dépasse la valeur du paramètre q, les deux autres relations ayant lieu, on peut envisager une extension de MLP (s, t, q) qui donnerait une augmentation du nombre de termes, avec

$$k_2 = \left]\frac{H}{q}\right[.$$

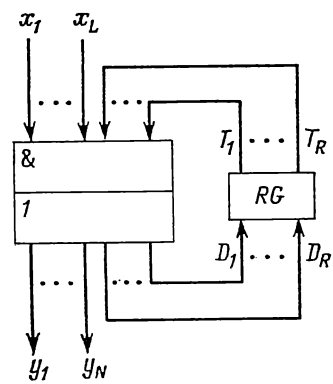


Fig. 11-6. Réalisation triviale d'un AMP avec une MLP (s, t, q)

Lorsque deux relations à la fois de (11-1), la deuxième et la troisième, ne sont pas vraies, on peut réaliser l'automate, en faisant appel à l'extension simultanée du nombre de sorties et de celui de termes de MLP (s, t, q). Signalons qu'une telle réalisation présente l'inconvénient de nécessiter trop de matériel; aussi les méthodes de synthèse d'AMP sur la base des MLP combinationnelles avec extension par sorties et par termes ne peuvent donc être conseillées que pour la réalisation d'automates relativement simples, aux valeurs des paramètres L, N, R et H peu élevées.

La construction d'AMP de grande dimension, dans les conditions où aucune des relations (11-1) n'est vérifiée, est loin d'être triviale, mais suscite en revanche nombre de difficultés.

Cherchons à réaliser un AMP avec des MLP combinationnelles dans un montage où les sorties de MLP (s, t, q) ne sont pas reliées

à ses entrées (fig. 11-7) [11]. On conçoit que dans ce cas le nombre de variables d'entrée $X(e_h)$ dans chaque terme e_h (le nombre de lettres dans chaque conjonction inscrite dans les lignes de la table de structure) ne doit pas dépasser $s - R$:

$$|X(e_h)| \leqslant s - R, \tag{11-2}$$

condition, en général, réalisable, même pour des automates micro-programmés de structure très complexe. Nous allons supposer que chaque terme e_h ($h = 1, \ldots, H$) apparaît dans au plus t fonctions

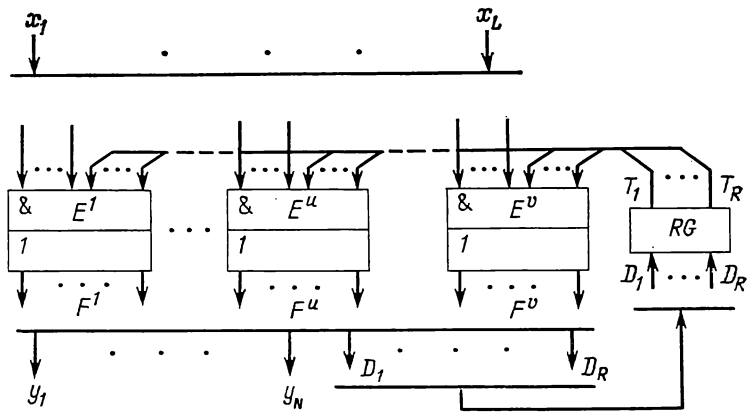


Fig. 11-7. Réalisation à un niveau d'un AMP avec une MLP (s, t, q)

de sortie et fonctions d'excitation de la mémoire de l'automate, i.e. que le nombre de micro-opérations et de fonctions d'excitation inscrites dans une ligne de sa table de structure ne dépasse pas t:

$$F(e_h)| \leqslant t. \tag{11-3}$$

Ici $F(e_h)$ désigne l'ensemble de fonctions auquel appartient le terme e_h. Le problème de synthèse d'AMP avec des MLP (s, t, q) se pose donc comme suit: sur l'ensemble $E = \{e_1, \ldots, e_H\}$ de termes de AMP vérifiant les conditions (11-2) et (11-3), chercher une partition $\pi_E = \{E^1, \ldots, E^U\}$, minimale quant au nombre de parties E^u et telle que chacune de ses parties vérifie les relations

$$|X(E^u)| \leqslant s - R; \tag{11-4}$$

$$|F(E^u)| \leqslant t; \tag{11-5}$$

$$|E^u| \leqslant q. \tag{11-6}$$

$X(E^u)$ de (11-4) est le nombre de variables d'entrée apparaissant dans tous les termes de la partie E^u:

$$X(E^u) = \bigcup_{e_h \in E^u} X(e_h).$$

$F(E^u)$ de (11-5) est l'ensemble des fonctions de sortie et des fonctions d'excitation de la mémoire de l'automate dans lesquelles apparaissent les termes de la partie E^u:

$$F(E^u) = \bigcup_{e_h \in E^u} F(e_h)$$

Nous allons ici décrire un algorithme approché séquentiel de recherche de la partition π_E des termes de l'AMP. L'exposé sera illustré sur l'exemple de synthèse de l'AMP donné par le tableau 10-3, les matrices utilisées seront des MLP (10, 10, 20). D'après le tableau 10-3, on construit le tableau 11-1 dans lequel à chaque terme e_h sont mis en correspondance son ensemble $X(e_h)$ des variables d'entrée et les ensembles $Y(e_h)$ et $D(e_h)$ des fonctions de sortie et des fonctions d'excitation de la mémoire où ce terme apparaît *). On tire immédiatement du tableau 11-1 qu'on ne peut pas choisir arbitrairement les combinaisons des termes à réaliser à l'aide d'une même MLP. Si, par exemple, on rapporte à une partie E^u les termes e_3, e_8, e_{21}, il y a violation de la condition (11-4), car $X(E^u) = \{x_1, x_2, x_3, x_4, x_6, x_{10}, x_{11}\}$. D'une façon analogue, l'inclusion dans une même partie E^p de termes e_2, e_6, e_{10} conduit à la non-observation de la condition (11-5), car dans ce cas $F(E^p) = \{y_1, y_2, y_{12}, y_3, y_5, y_{13}, y_{18}, D_1, D_2, D_3, D_4\}$, bien que la relation (11-4) soit vérifiée: $X(E^p) = \{x_1, x_6, x_7, x_9\}$.

Dans notre algorithme nous allons nous guider sur trois critères principaux:

1. $c(e_i, e_j)$, proximité de deux termes e_i et e_j;
2. $c(e_i, E^u)$, proximité du terme e_i et de la partie E^u;
3. $w(e_i, E^u)$, fonction de poids du terme e_i par rapport à la partie $E^u \subset E$. Ici e_i est le terme qu'on n'a pas inclu dans la partie E^u ($e_i \in E \setminus E^u$), et $w(e_i, E^u)$ constitue le critère utilisé pour juger s'il faut ou non inclure le terme examiné dans la partie E^u de la partition π_E.

On définit la proximité de deux termes par l'expression

$$c(e_i, e_j) = k_1 \mid X(e_i) \cap X(e_j) \mid + k_2 \mid Y(e_i) \cap Y(e_j) \mid + \\ + k_3 \mid D(e_i) \cap D(e_j) \mid, \qquad (11\text{-}7)$$

*) Nous n'avons retenu, pour l'inscrire dans le tableau 11-1, qu'un élément, notamment y_7, y_8, y_{10} et y_{16}, des deux formant chacun des ensembles $\{y_7, y_{11}\}$, $\{y_8, y_9\}$, $\{y_{10}, y_{15}\}$ et $\{y_{16}, y_{17}\}$; la raison en est que, dans l'exemple considéré, les micro-opérations composant ces ensembles y sont toujours présentes simultanément (voir tabl. 10-10).

Tableau 11-1

Correspondance entre les termes, les variables et les fonctions de l'AMP

e_h	$X(e_h)$	$Y(e_h)$	$D(e_h)$
e_1	x_1, x_2	y_1, y_2	—
e_2	x_1	y_1, y_2, y_{12}	D_4
e_3	x_1, x_2, x_3	y_2, y_{12}	D_1, D_3
e_4	x_1, x_2, x_3	y_1, y_2	D_1, D_2, D_3
e_5	x_6	—	D_2, D_3, D_4
e_6	x_6	y_3, y_5	D_1, D_2, D_3
e_7	x_{10}	y_3, y_5	D_1, D_2, D_3
e_8	x_4, x_{10}	y_5, y_8	D_2, D_4
e_9	x_4, x_{10}	y_{10}	D_2, D_3
e_{10}	x_7, x_9	y_{13}, y_{18}	D_1, D_2, D_4
e_{11}	x_7	y_{13}	D_1, D_3
e_{12}	x_7, x_9	y_{13}, y_{14}	D_2, D_4
e_{13}	x_1	y_7	D_4
e_{14}	x_1	y_{16}	D_1, D_3
e_{15}	x_5	y_1, y_2	—
e_{16}	x_5	y_{16}	—
e_{17}	x_8	y_{14}, y_{18}	D_2, D_3, D_4
e_{18}	x_8	y_{13}, y_{18}	D_1, D_2, D_4
e_{19}	x_9	y_6	D_1, D_2, D_4
e_{20}	x_9	y_4, y_6	D_3
e_{21}	x_6, x_{11}	y_5, y_8	D_2, D_3, D_4
e_{22}	x_{11}	y_3, y_8	D_1, D_2, D_3
e_{23}	x_6, x_{11}	y_{10}	D_2, D_3

où k_1, k_2, k_3 sont les coefficients de pondération. Leur rapport est défini par le degré d'importance des restrictions (11-4) à (11-6), i.e. des relations entre les paramètres L, N et R qui sont le nombre des variables d'entrée, ceux des fonctions de sortie et des fonctions d'excitation de l'automate, et les paramètres s et t, nombre d'entrées et celui de sorties d'une MLP (s, t, q). Posons $k_1 = k_2 = k_3 = 1$. Dans notre exemple $c(e_3, e_4) = |\{x_1, x_2, x_3\} \cap \{x_1, x_2, x_3\}| + |\{y_2, y_{12}\} \cap \{y_1, y_2\}| + |\{D_1, D_3\} \cap \{D_1, D_2, D_3\}| = |\{x_1, x_2, x_3\}| + |\{y_2\}| + |\{D_1, D_3\}| = 3 + 1 + 2 = 6$. Autrement dit, les lignes 3 et 4 du tableau 11-1 (et aussi du tableau 10-3) ont en commun trois variables d'entrée, une fonction de sortie et deux fonctions d'excitation.

La proximité du terme e_i de la partie E^u est donnée par l'expression :

$$c(e_i, E^u) = \sum_{e_j \in E^u} c(e_i, e_j). \tag{11-8}$$

Enfin, la fonction de poids $w(e_i, E^u)$ peut être calculée par la formule :

$$w(e_i, E^u) = c(e_i, E^u) - c(e_i, E \setminus E^u), \tag{11-9}$$

qui exprime la différence entre les proximités du terme e_i par rapport à la partie E^u et à l'ensemble $E \setminus E^u$ des termes non compris dans E^u.

Selon l'algorithme de recherche de la partition π_E, la formation de chaque partie E^u $(u = 1, \ldots, U)$ s'effectue en deux étapes. A la première étape on inclut dans E^u l'élément de base, le terme e_i, tel que

$$c(e_i, E' \setminus \{e_i\}) = \max_j c(e_j, E' \setminus \{e_j\}),$$

où $E' = E \setminus \left(\bigcup_{m=1}^{u-1} E^m\right)$ est l'ensemble des termes qu'on n'a pas encore rapportés à une partie de la partition, $e_i, e_j \in E'$, et $c(e_j, E' \setminus \{e_j\})$ se calcule par la formule (11-8). On voit donc qu'en qualité d'élément de base (premier) d'une partie E^u il convient de prendre le terme le plus proche de tous les termes non répartis, i.e. des termes non inclus dans les parties $E^1, \ldots, E^{u-1}$.

La deuxième étape est itérée. A chaque pas on ajoute à la partie E^u le terme suivant, se conformant à la règle décrite ci-dessous. Le processus de formation d'une partie E^u prend fin s'il ne reste plus de termes à examiner, i.e. $E \setminus \left(\bigcup_{m=1}^{u} E^m\right) = \varnothing$, ou si aucun terme ne peut être ajouté à la partie E^u sans que soit violée l'une quelconque des restrictions (11-4) à (11-6).

La règle de recherche du terme à ajouter à la partie E^u se résume à ce qui suit. Supposons qu'après le i-ième pas on ait $E^u = (e_\alpha, e_\beta, \ldots, e_\omega)$. Au $(i+1)$-ième pas on calcule par la formule (11-9) la fonction de poids $w(e_t, E^u)$ pour chacun des termes non répartis $e_t \in E \setminus \left(\bigcup_{m=1}^{u} E^u\right)$ tel que $c(e_t, E^u) \neq 0$. De tous ces termes on choisit le terme e_j tel que $w(e_j, E^u) = \max_t w(e_t, E^u)$, si seulement son adjonction à la partie E^u n'entraîne pas de violation de l'une quelconque des restrictions (11-4) à (11-6). Si plusieurs termes répondent à ces critères, on en choisit un qui présente $\max_t c(e_t, E^u)$; lorsque ce critère est égal pour plus d'un terme, on prend l'un quelconque de ces termes, sous ce rapport équivalents.

Si les conditions (11-4) à (11-6) s'opposent à l'inclusion dans la partie E^u du terme e_j présentant une fonction de poids $w(e_j, E^u)$ maximale, il convient d'examiner, par ordre de décroissance de leur fonction de poids, les termes restants, en ne retenant que ceux dont la proximité de la partie E^u est non nulle. Parmi ces termes, on choisit le premier, conforme aux conditions (11-4) à (11-6). On examine en dernier, toujours dans le même ordre, les termes dont la proximité de la partie E^u est nulle: $c(e_t, E^u) = 0$.

L'algorithme séquentiel de recherche de la partition π_E comprendra donc les opérations suivantes:

1. Former la matrice d'adjacence de toutes les paires de termes de l'ensemble E, en inscrivant la fonction de proximité $c(e_i, e_j)$ de deux éléments $e_i, e_j \in E$ dans la case se trouvant à l'intersection de la ligne i et de la colonne j.
2. Poser $u = 1$, $E' = E$.
3. Choisir un élément de base, en procédant comme décrit ci-dessus, pour amorcer la formation de la partie u-ième E^u de la partition cherchée π_E.
4. Former la partie E^u de la partition π_E.
5. Si $E' \setminus E^u = \varnothing$, aller en 8, sinon aller en 6.
6. Poser $E' = E' \setminus E^u$ et $u = u + 1$.
7. Calculer les fonctions $c(e_i, E' \setminus \{e_i\})$ pour tous les $e_i \in E'$ et aller en 3.
8. Fin.

Illustrons le « travail » de l'algorithme proposé sur l'exemple de synthèse de l'AMP donné par le tableau 10-3 que nous allons réaliser avec des MLP (10, 10, 20). Les données initiales nécessaires pour calculer les fonctions $c(e_i, e_j)$, $c(e_i, E^u)$ et $W(e_i, E^u)$ sont réunies dans le tableau 11-1. La matrice d'adjacence des termes, compte tenu de $k_1 = k_2 = k_3 = 1$, est représentée par le tableau 11-2, dont les lignes et les colonnes portent les numéros des termes de l'ensemble E (le numéro i correspond au terme e_i). Le tableau 11-3 comporte toutes les données de départ et montre la marche des

Tableau 11-2

Matrice de proximité des termes

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	—	3	3	4	0	0	0	0	0	0	0	0	1	1	2	0	0	0	0	0	0	0	0
2	3	—	3	3	1	0	0	1	0	1	0	1	2	1	2	0	1	1	1	0	1	0	0
3	3	3	—	6	1	2	2	0	1	1	2	0	1	3	1	0	1	1	1	1	1	2	0
4	4	3	6	—	2	3	3	1	2	2	2	1	1	3	2	0	2	2	2	1	2	3	2
5	0	1	1	2	—	3	2	2	2	2	1	2	1	1	0	0	3	2	2	1	4	2	3
6	0	0	2	3	3	—	5	2	2	2	2	1	0	2	0	0	2	2	2	1	4	4	3
7	0	0	2	3	2	5	—	3	3	2	2	1	0	1	0	0	2	2	2	1	3	4	2
8	0	1	0	1	2	2	3	—	3	2	1	2	1	0	0	0	2	2	2	0	4	2	1
9	0	0	1	2	2	2	3	3	—	1	1	1	0	1	0	0	2	1	1	1	2	2	3
10	0	1	1	2	2	2	2	2	1	—	3	5	1	1	0	0	3	5	4	1	2	2	1
11	0	0	2	2	1	2	2	1	1	3	—	2	0	2	0	0	1	2	1	1	1	2	1
12	0	1	0	1	2	1	1	2	1	5	2	—	1	0	0	0	2	3	3	1	2	1	1
13	1	2	1	1	1	0	0	1	0	1	0	1	—	1	0	0	1	1	1	0	1	0	1
14	1	1	3	3	1	2	1	0	1	1	2	0	1	—	0	1	1	1	1	1	1	2	1
15	2	2	1	2	0	0	0	0	0	0	0	0	0	0	—	1	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	—	0	0	0	0	0	0	0
17	0	1	1	2	3	2	2	2	2	3	1	2	1	1	0	0	—	4	2	1	3	2	2
18	0	1	1	2	2	2	2	2	1	5	2	3	1	1	0	0	4	—	3	0	2	2	1
19	0	1	1	2	2	2	2	2	1	4	1	3	1	1	0	0	2	3	—	2	2	2	1
20	0	0	1	1	1	1	1	0	1	1	1	1	0	1	0	0	1	0	2	—	1	1	1
21	0	1	1	2	4	4	3	4	2	2	1	2	1	1	0	0	3	2	2	1	—	4	4
22	0	0	2	3	2	4	4	2	2	2	2	1	0	2	0	0	2	2	2	1	4	—	3
23	0	0	0	2	3	3	2	1	3	1	1	1	0	1	0	0	2	1	1	1	4	3	—

Tableau 11-3

Formation des ensembles de la partition π_E

E	$c(e_i, E')$	$w(e_i, E^1)$ 1	2	3	4	5	6	7	8	9	10	11	$c(e_i, E')$	$w(e_i, E^2)$ 1	2	3	4	5	6	7	8	9	10	$c(e_i, E')$	$w(e_i, E^3)$ 1	2	3	4
e_1	14	−6	−6	−2	⊕	+	+	+	+	+	+	+																
e_2	22	−16	−16	−12	−6	⊕	+	+	+	+	+	+																
e_3	33	−21	−21	−19	−13	−7	−5	⊕	+	+	+	+																
e_4	49	⊕	+	+	+	+	+	+	+	+	+	+																
e_5	37	−33	−33	−33	−33	−31	−29	−27	−25	−23	⊕	+																
e_6	42	−36	−36	−36	−36	−36	−36	−32	−28	−24	−18	−	30	−22	−20	−14	−10	−6	⊕	+	+	+	+					
e_7	40	−30	−30	−30	−30	−30	−30	−26	−24	−20	−	−	30	−24	−22	−18	−12	−6	4	⊕	+	+	+					
e_8	31	−29	−29	−29	−29	−27	−25	−25	−25	−23	−19	−	25	−17	−17	−15	−9	⊕	+	+	+	+	+					
e_9	29	−25	−25	−25	−25	−25	−25	−23	−21	−19	−	−	24	−20	−18	−12	⊕	+	+	+	+	+	+					
e_{10}	41	−37	−37	−37	−37	−35	−33	−31	−29	−23	−19	−	30	−26	−24	−22	−20	−16	−12	−8	−4	4	−	13	−3	⊕	+	+
e_{11}	27	−23	−23	−23	−23	−23	−23	−19	−15	⊕	+	+																
e_{12}	30	−28	−28	−28	−28	−26	−24	−24	−24	−20	−	−	25	−21	−19	−17	−15	−11	−9	−7	−5	1	−	12	−6	4	⊕	+
e_{13}	14	−12	−10	−10	−8	−4	⊕	+	+	+	+	+																
e_{14}	26	−20	−18	−18	−16	−14	−12	−6	⊕	+	+	+																
e_{15}	8	−4	−4	⊕	+	+	+	+	+	+	+	+																
e_{16}	2	−2	⊕	+	+	+	+	+	+	+	+	+																
e_{17}	37	−33	−33	−33	−33	−31	−29	−27	−25	−23	−17	−	27	−21	−19	−15	−11	−7	−3	1	5	−	−	11	−3	3	7	⊕
e_{18}	37	−33	−33	−33	−33	−31	−29	−27	−25	−21	−	−	29	−25	−25	−23	−21	−17	−13	−9	−5	1	−	14	⊕	+	+	+
e_{19}	35	−31	−31	−31	−31	−29	−27	−25	−23	−21	−	−	28	−24	−20	−18	−16	−12	−8	−4	0	⊕	+					
e_{20}	16	−14	−14	−14	−14	−14	−14	−12	−10	−	−	−	13	−11	⊕	+	+	+	+	+	+	+	+					
e_{21}	44	−40	−40	−40	−40	−38	−38	−36	−34	−32	−24	−	34	⊕	+	+	+	+	+	+	+	+	+					
e_{22}	40	−36	−36	−36	−36	−36	−36	−32	−28	−24	−20	−	30	−22	−20	−14	−10	−6	2	10	⊕	+	+					
e_{23}	30	−26	−26	−26	−26	−26	−26	−26	−24	−22	−	−	26	−18	−16	⊕	+	+	+	+	+	+	+					
X		$x_1, x_2, x_3, x_5, x_7, x_6$												$x_6, x_{11}, x_9, x_4, x_{10}$											x_8, x_7, x_9			
Y		$y_1, y_2, y_{16}, y_{12}, y_7, y_{13}$												$y_5, y_8, y_4, y_6, y_{10}, y_3$											y_{13}, y_{18}, y_{14}			
D		D_1, D_2, D_3, D_4												D_2, D_3, D_4, D_1											D_1, D_2, D_4, D_3			

opérations de formation des parties de la partition π_E. Dans la première colonne du tableau sont énumérés tous les termes $e_h \in E$. Les colonnes 2, 14 et 25 sont réservées aux valeurs des fonctions de proximité de chacun des termes par rapport à l'ensemble des termes non répartis, à l'étape en cours, entre les parties de la partition. D'après notre algorithme, ces quantités servent de critère de choix des éléments de base des parties à former. Les colonnes 3 à 13, 15 à 24 et 26 à 29 comportent les quantités $w(e_i, E^u)$ qui sont utilisées pour l'adjonction du terme suivant à la partie en cours de formation. Un « plus » inscrit dans la ligne se rapportant

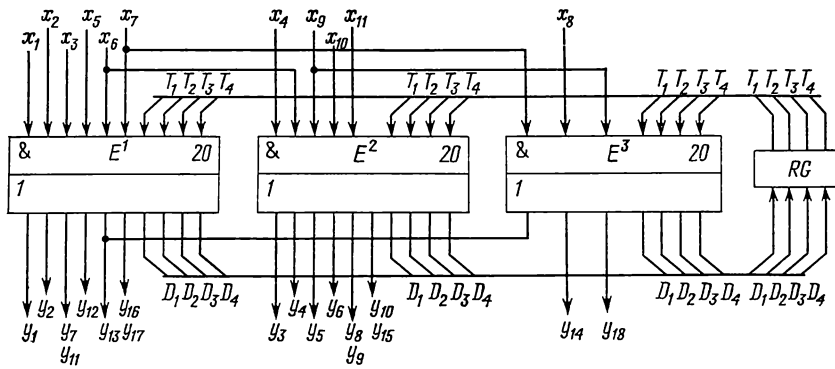


Fig. 11-8. Schéma logique d'un AMP réalisé avec des MLP (10, 10, 20)

à la formation de la u-ième partie signifie que le terme correspondant à cette ligne est rapporté à cette partie. Par exemple, le terme e_2 est rapporté à la première partie de la partition au cinquième pas de la procédure de sa formation. Un « moins » dans une ligne signifie que le terme correspondant ne peut pas être ajouté à la partie en cours de formation en raison des contraintes (11-4) à (11-6). Ainsi, par exemple, on est obligé à éliminer le terme e_{20} comme candidat à la partie E^1 au neuvième pas pour cause de violation de la contrainte (11-5). Pour faciliter le contrôle de l'observation des conditions (11-4) à (11-6), on donne dans les trois lignes en bas du tableau 11-3 les variables d'entrée des termes de la partie en cours de formation, ainsi que les fonctions de sortie et les fonctions d'excitation dont font partie les termes en question.

Dans notre cas, la partition π_E se compose de trois parties:

$$E^1 = \{e_1, e_2, e_3, e_4, e_5, e_{11}, e_{13}, e_{14}, e_{15}, e_{16}\};$$
$$X(E^1) = \{x_1, x_2, x_3, x_5, x_6, x_7\};$$
$$Y(E^1) = \{y_1, y_2, y_7, y_{12}, y_{13}, y_{16}\};$$
$$D(E^1) = \{D_1, D_2, D_3, D_4\}.$$

Tableau 11-4

Programmation de la MLP_1 (10, 10, 20)

n^{os}	T_1	T_2	T_3	T_4	x_1	x_2	x_3	x_5	x_6	x_7	Termes	y_1	y_2	y_7	y_{12}	y_{13}	y_{16}	D_1	D_2	D_3	D_4
	Entrées											Sorties									
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	1	0	—	—	—	—	e_1	1	1	·	·	·	·	·	·	·	·
2	0	0	0	0	0	—	—	—	—	—	e_2	1	1	·	1	·	·	·	·	·	1
3	0	0	0	0	1	1	0	—	—	—	e_3	·	1	·	1	·	·	1	·	1	·
4	0	0	0	0	1	1	1	—	—	—	e_4	1	1	·	·	·	·	1	1	1	·
5	0	1	1	1	—	—	—	—	1	—	e_5	·	·	·	·	·	·	·	1	1	1
6	1	1	0	1	—	—	—	—	—	1	e_{11}	·	·	·	·	1	·	1	·	1	·
7	0	0	0	1	0	—	—	—	—	—	e_{13}	·	·	1	·	·	·	·	·	·	1
8	0	0	0	1	1	—	—	—	—	—	e_{14}	·	·	·	·	·	1	1	·	1	·
9	1	0	1	0	—	—	—	1	—	—	e_{15}	1	1	·	·	·	·	·	·	·	·
10	1	0	1	0	—	—	—	0	—	—	e_{16}	·	·	·	·	·	1	·	·	·	·
11	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·	·	·	·	·	·	·
...											...										
20	—	—	—	—	—	—	—	—	—	—	—	·							·	·	·

Tableau 11-5

Programmation de la MLP_2 (10, 10, 20)

n^{os}	T_1	T_2	T_3	T_4	x_4	x_6	x_9	x_{10}	x_{11}	—	Termes	y_3	y_4	y_5	y_6	y_8	y_{10}	D_1	D_2	D_3	D_4
	Entrées											Sorties									
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	—	0	—	—	—	—	e_6	1	·	1	·	·	·	1	1	1	·
2	1	1	1	0	—	—	—	1	—	—	e_7	1	·	1	·	·	·	1	1	1	·
3	1	1	1	0	0	—	—	0	—	—	e_8	·	·	1	·	1	·	·	1	·	1
4	1	1	1	0	1	—	—	0	—	—	e_9	·	·	·	·	·	1	·	1	1	·
5	0	1	0	1	—	—	0	—	—	—	e_{19}	·	·	·	1	·	·	1	1	·	1
6	0	1	0	1	—	—	1	—	—	—	e_{20}	·	1	·	1	·	·	·	·	1	·
7	0	1	1	0	—	0	—	—	1	—	e_{21}	·	·	1	·	1	·	·	1	1	1
8	0	1	1	0	—	—	—	—	0	—	e_{22}	1	·	·	·	1	·	1	1	1	·
9	0	1	1	0	—	1	—	—	1	—	e_{23}	·	·	·	·	·	1	·	1	1	·
10	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·	·	·	·	·	·	·
...											...										
20	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·	·	·	·		·	·

$$E^2 = \{e_6, e_7, e_8, e_9, e_{19}, e_{20}, e_{21}, e_{22}, e_{23}\};$$
$$X(E^2) = \{x_4, x_6, x_9, x_{10}, x_{11}\};$$
$$Y(E^2) = \{y_3, y_4, y_5, y_6, y_8, y_{10}\};$$
$$D(E^2) = \{D_1, D_2, D_3, D_4\}.$$
$$E^3 = \{e_{10}, e_{12}, e_{17}, e_{18}\};$$
$$X(E^3) = \{x_7, x_8, x_9\};$$
$$Y(E^3) = \{y_{13}, y_{14}, y_{18}\};$$
$$D(E^3) = \{D_1, D_2, D_3, D_4\}.$$

Ces expressions conduisent au circuit logique de la figure 11-8. On a ajouté les micro-opérations y_{11}, y_{17}, y_9 et y_{15} aux sorties y_7, y_{16}, y_8 et y_{10} de MLP respectivement, car, comme on l'a men-

Tableau 11-6

Programmation de la MLP$_3$ (10, 10, 20)

n^{os}	T_1	T_2	T_3	T_4	x_7	x_8	x_9	—	—	—	Termes	y_{13}	y_{14}	y_{18}	D_1	D_2	D_3	D_4	—	—	—
	Entrées											Sorties									
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	1	1	0	1	0	—	1	—	—	—	e_{10}	1	·	1	1	1	·	1	·	·	·
2	1	1	0	1	0	—	0	—	—	—	e_{12}	1	1	·	·	1	·	1	·	·	·
3	0	0	1	0	—	1	—	—	—	—	e_{17}	·	1	1	·	1	1	1	·	·	·
4	0	0	1	0	—	0	—	—	—	—	e_{18}	1	·	1	1	1	·	1	·	·	·
5	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·	·	·	·	·	·	·
...											...										
20	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·						·

tionné au début des raisonnements, les couples en question se rencontrent toujours ensemble. Les tables de programmation de chacune des MLP, analogues au tableau 10-1, sont résumées dans les tableaux 11-4, 11-5 et 11-6.

11-3. Réalisation triviale de l'automate microprogrammé avec des MLP à mémoire

Rappelons qu'à la différence d'une MLP (s, t, q), une MLP à mémoire comporte un registre interne RG de liaison entre les matrices M_2 et M_1 (voir fig. 11-2, *a*). Le nombre r de positions binaires du registre de MLP (s, t, q, r) est choisi tel que $r \geqslant \geqslant]\log_2 q[$. Ainsi, un AMP quelconque dont les nombres L, N et H

de variables d'entrée, de micro-opérations et de lignes dans la table des transitions respectivement vérifient les conditions

$$L \leqslant s, \quad N \leqslant t, \quad H \leqslant q \tag{11-10}$$

peut être réalisé d'une façon triviale au moyen d'une seule MLP (s, t, q, r). Toutes les s entrées et t sorties de la MLP (s, t, q, r) peuvent dans ce cas être utilisées par les variables d'entrée et les micro-opérations de l'AMP, tandis que lors d'une réalisation de l'automate avec des MLP (s, t, q) il fallait prévoir quelque R entrées et R sorties de MLP pour établir des liaisons avec les éléments de mémoire de l'AMP, lesquels nécessitaient des circuits supplémentaires. On conçoit que dans certains cas les matrices MLP (s, t, q, r) présentent des avantages par rapport aux matrices MLP (s, t, q).

Si une au moins des conditions (11-10) n'est pas vérifiée, on aura besoin pour réaliser un AMP, de plusieurs MLP (s, t, q, r) associées dans un montage approprié. Pour un automate avec $N > t$, mais $L \leqslant s$ et $H \leqslant q$, on peut envisager une extension k-uple du nombre de sorties d'une MLP (s, t, q, r), $k = \left]\frac{N}{t}\right[$, qui consiste, comme dans le cas des MLP (s, t, q), à connecter les entrées respectives de toutes les k MLP (s, t, q, r). Les matrices M_1 des k MLP (s, t, q, r) élaborent les mêmes termes (au nombre $H \leqslant q$), et les kt sorties des matrices M_2 fournissent l'ensemble $Y = \{y_1, \ldots, y_N\}$ des micro-opérations de l'AMP. L'ensemble Y se laisse décomposer d'une façon arbitraire en $k = \left]\frac{N}{t}\right[$ sous-ensembles (parties) contenant chacun au plus t éléments. Les micro-opérations de chaque sous-ensemble sont réalisées au moyen de l'une des k MLP (s, t, q, r).

Soit à réaliser, avec des MLP (10, 10, 20, 5), un automate de paramètres $L = 10$, $N = 18$ et $H = 20$ donné par sa table de structure A. Une extension du nombre de sorties s'impose. A cet effet, décomposons l'ensemble des micro-opérations $Y = \{y_1, \ldots, y_{18}\}$ en deux sous-ensembles $Y^1 = \{y_1, \ldots, y_{10}\}$ et $Y^2 = \{y_{11}, \ldots, y_{18}\}$. Imaginons deux nouvelles tables de structure, tabl. B et tabl. C, qui ne diffèrent de la table A et entre elles que par le contenu de la colonne Y_t. Nous inscrirons dans la colonne Y_t de la table B les micro-opérations de l'ensemble Y^1 et dans la table C, celles de l'ensemble Y^2. Les micro-opérations qu'on inscrira dans la colonne Y_t des tables B et C devront l'être dans les mêmes lignes que dans la table A. Si, par exemple, à une transition quelconque de la table A correspondent les micro-opérations y_1, y_3, y_{12}, on inscrira dans la colonne Y_t de la table B, sur la ligne correspondante, y_1, y_3, et dans la case identique de la table C, on mettra y_{12}. On obtient de la sorte deux automates presque identiques, dont les tables de structure B et C ne diffèrent que par le contenu de leur colonne de signaux de sortie. Si l'application d'un signal d'entrée

X_h produit la transition de l'un des automates de l'état a_m dans l'état a_s accompagnée de l'apparition de la micro-instruction $Y_t^1 \subseteq \subseteq Y^1$ en sa sortie, le même signal X_h produira la transition (a_m, a_s) dans l'autre automate, avec apparition de la micro-instruction $Y_t^2 \subseteq Y^2$. Il est clair que chacun de ces automates peut être réalisé d'une façon triviale avec une seule MLP (10, 10, 20, 5) directement d'après les tables de structure B et C.

Si l'on se propose de réaliser, avec des MLP (s, t, q, r), un automate dont soit le nombre de variables d'entrée dépasse celui d'entrées du circuit dont on dispose $(L > s)$, soit le nombre de lignes de la table de structure est supérieur à celui des termes de MLP $(H > > q)$, on devra auparavant décomposer l'automate initial en des automates plus simples, susceptibles d'être réalisés avec une seule MLP chacun.

11-4. Décomposition de l'automate microprogrammé

Rappelons qu'on entend par décomposition d'un automate la construction pour l'automate donné S d'un réseau C composé d'automates interdépendants. A chaque réseau C on peut mettre en correspondance son automate résultant S_C. On dit que le réseau C est le résultat de décomposition de l'automate S si S_C possède un sous-automate isomorphe à S.

Au chapitre quatre nous avons traité des méthodes algébriques de décomposition des automates abstraits. On connaît des tentatives [30] d'appliquer ces méthodes à la décomposition d'automates microprogrammés donnés par des tables de transitions du type du tableau 8-1. On s'est heurté à nombre de problèmes dont on n'entrevoit pas encore la solution. Tout d'abord, la décomposition des AMP complique énormément le problème du choix de l'ensemble initial de partitions orthogonales, problème qui est déjà très compliqué même pour les automates abstraits avec un nombre peu élevé de signaux d'entrée, tandis que dans le cas de l'AMP on a affaire à des conjonctions; si une conjonction est composée de k lettres et l'AMP a L variables d'entrée, il correspond à la conjonction en question 2^{L-k} signaux d'entrée. Ensuite, on ne sait pas encore comment construire un réseau dont la table de transitions de chacun des automates composants ne soit pas aussi volumineuse que celle de l'automate qu'on se propose de décomposer. Or, la longueur de la table de transitions est le critère des plus importants de la complexité du schéma logique de l'AMP. Enfin, reste le problème de réalisation optimale de la fonction de sortie du réseau.

Dans l'ouvrage [12] est proposée une méthode simple de décomposition de l'AMP *) qui part des considérations suivantes:

* Une méthode analogue de décomposition d'automates abstraits est considérée dans l'ouvrage [13].

1) le réseau cherché doit se composer d'automates dont la longueur totale des tables de transitions est à peu près égale à la longueur de la table de transitions de l'automate initial;

2) le réseau doit être formé d'automates composants séquentiels.

La procédure de décomposition d'un automate microprogrammé de Mealy sera illustrée sur l'exemple de l'automate donné par le tableau 10-3.

Supposons définie sur l'ensemble A des états internes de l'automate à décomposer $S = (A, X, Y, \delta, \lambda, a_1)$ une partition $\pi = \{A^1, \ldots, A^n\}$. Nous reviendrons sur la question du choix de la

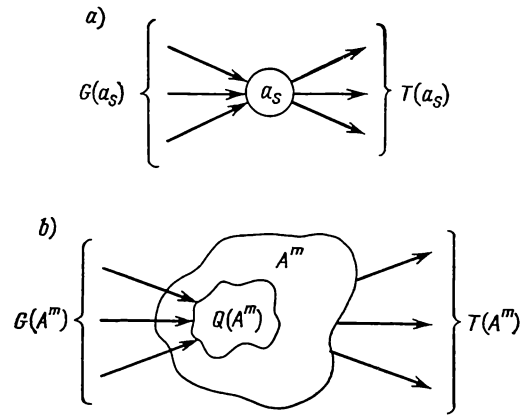


Fig. 11-9. Diverses transitions à partir d'un état interne et vers celui-ci (a); la même chose pour un ensemble de partition (b)

partition après avoir terminé de l'exposé de la procédure de décomposition. Dans notre exemple, posons $\pi = \overline{1.5.6}, \overline{2.3.9}, \overline{4.7.8}$, i.e. $A^1 = \{a_1, a_5, a_6\}$, $A^2 = \{a_2, a_3, a_9\}$, $A^3 = \{a_4, a_7, a_8\}$. Comme auparavant, $X(a_s)$ désignera l'ensemble des variables d'entrée qui apparaissent dans la colonne X_h de la table de transitions de l'AMP dans toutes les transitions à partir de l'état a_s. De même, $Y(a_s)$ est l'ensemble des micro-opérations de la colonne Y_t correspondant aux transitions de l'état a_s. Désignons par $G(a_s)$, l'ensemble de tous les états, $a_s \notin G(a_s)$, à partir desquels existent des transitions dans l'état a_s, et par $T(a_s)$, l'ensemble des états dans lesquels il est possible de passer de l'état a_s, $a_s \notin T(a_s)$ (fig. 11-9, a):

$$G(a_s) = \{a_i \mid \delta(a_i, X_h) = a_s,\ i \neq s\};$$
$$T(a_s) = \{a_j \mid \delta(a_s, X_f) = a_j,\ j \neq s\}.$$

Pour l'automate donné par le tableau 10-3 on a

$$X(a_3) = \{x_4, x_{10}\};\ Y(a_3) = \{y_3, y_5, y_8, y_9, y_{10}, y_{15}\};$$
$$G(a_3) = \{a_1, a_2, a_9\};\ T(a_3) = \{a_8, a_9\}.$$

Introduisons des désignations analogues pour le sous-ensemble A^m de la partition π:

$$X(A^m) = \bigcup_{a_s \in A^m} X(a_s);$$

$$Y(A^m) = \bigcup_{a_s \in A^m} Y(a_s).$$

Avec les notations adoptées, $X(A^m)$ et $Y(A^m)$ sont les ensembles des variables d'entrée et des variables de sortie inscrites dans la table de transitions de l'automate dans toutes les transitions et à partir de tous les états figurant dans A^m.

Soit $G(A^m)$ l'ensemble des états, non appartenant à A^m, à partir desquels il y a des transitions dans les états de A^m, $T(A^m)$ l'ensemble des états, non appartenant à A^m, vers lesquels il y a des transitions des états de A^m, $Q(A^m)$ l'ensemble des états de A^m vers lesquels il y a des transitions à partir des états appartenant aux parties de π autres que A^m (états de l'ensemble $G(A^m)$):

$$G(A^m) = \{a_i \notin A^m \mid \delta(a_i, X_h) = a_s;\ a_s \in A^m\};$$

$$T(A^m) = \{a_j \notin A^m \mid \delta(a_s, X_f) = a_j;\ a_s \in A^m\};$$

$$Q(A^m) = \{a_s \in A^m \mid \delta(a_i, X_h) = a_s;\ a_i \notin A^m\}.$$

On a réuni dans le tableau 11-7 tous ces ensembles pour chacune des parties de la partition π que nous avons définie plus haut.

Tableau 11-7

Transitions entre les ensembles de la partition

A^m	$Q(A^m)$	$G(A^m)$	$T(A^m)$
a_1, a_5, a_6	a_6	a_4	a_3
a_2, a_3, a_9	a_2, a_3	a_1, a_7	a_8
a_4, a_7, a_8	a_8	a_3	a_2, a_6

A l'automate S et à la partition π on met en correspondance un réseau d'automates $S^1, \ldots, S^n$ dont le nombre doit donc être égal à celui de parties de la partition π. Dans notre exemple, le réseau se composera de trois AMP: S^1, S^2 et S^3.

L'automate $S^m = (B^m, X^m, Y^m, \delta^m, \lambda^m, a_{m1})$, $m = 1, \ldots, n$, sera défini comme suit.

1. Formons l'ensemble B^m des états de S^m en ajoutant à la partie A^m de la partition π un état supplémentaire b_m: $B^m = A^m \cup \cup \{b_m\}$. Le nombre d'états de chaque automate composant sera donc d'une unité supérieur au nombre d'états rapportés à la partie correspondante de la partition π. Dans notre exemple:

$$B^1 = \{b_1, a_1, a_5, a_6\};$$
$$B^2 = \{b_2, a_2, a_3, a_9\};$$
$$B^3 = \{b_3, a_4, a_7, a_8\}.$$

2. Les variables d'entrée de l'automate S^m seront définies comme suit:

$$X^m = X(A^m) \cup Z_x^m. \tag{11-11}$$

Ici Z_x^m sont des variables supplémentaires nécessaires pour assurer la liaison de l'entrée de l'automate S^m avec les autres automates du réseau. Pour déterminer Z_x^m on met en correspondance à chaque état $a_s \in Q(A^m)$ une variable d'entrée z_s, de même indice s, de l'automate microprogrammé S^m. Alors, ainsi qu'on le montrera plus bas, si l'automate initial qu'on cherche à décomposer possède une transition de $a_i \notin A^m$ dans $a_s \in A^m$, l'automate composant qui admet l'état a_i produira le signal de sortie (micro-opération) z_s qui fera prendre à l'automate S^m l'état a_s:

$$Z_x^m = \{z_s \mid \delta(a_i, X_h) = a_s;\ a_s \in A^m,\ a_i \notin A^m\}.$$

La colonne $Q(A^m)$ du tableau 11-7 nous donne: $Z_x^1 = \{z_6\}$; $Z_x^2 = \{z_2, z_3\}$; $Z_x^3 = \{z_8\}$. Conformément à (11-11) on a:

$$X^1 = \{x_1, x_2, x_3, x_5, z_6\};$$
$$X^2 = \{x_4, x_6, x_{10}, x_{11}, z_2, z_3\};$$
$$X^3 = \{x_7, x_8, x_9, z_8\}.$$

3. Les variables de sortie de l'automate composant S^m seront définies comme suit:

$$Y^m = Y(A^m) \cup Z_y^m. \tag{11-12}$$

Ici Z_y^m sont les variables supplémentaires nécessaires pour assurer la liaison de la sortie de l'automate S^m avec les autres automates composants. Pour déterminer Z_y^m on met en correspondance à chaque état $a_j \in T(A^m)$ une variable de sortie z_j, de même indice j, de l'automate S^m:

$$Z_y^m = \{z_j \mid \delta(a_s, X_f) = a_j;\ a_s \in A^m,\ a_j \notin A^m\}.$$

La colonne $T(A^m)$ du tableau 11-7 nous donne immédiatement: $Z_y^1 = \{z_3\}$; $Z_y^2 = \{z_8\}$; $Z_y^3 = \{z_2, z_6\}$. Conformément à (11-12) on a:

$$Y^1 = \{y_1, y_2, y_7, y_{11}, y_{12}, y_{16}, y_{17}, z_3\};$$
$$Y^2 = \{y_3, y_5, y_8, y_9, y_{10}, y_{15}, z_8\};$$
$$Y^3 = \{y_4, y_6, y_{13}, y_{14}, y_{18}, z_2, z_6\}.$$

4. Définissons les fonctions de transition et de sortie de l'automate composant S^m. Supposons que dans l'automate initial S l'application d'un signal d'entrée X_h produit une transition de a_i dans a_j et l'apparition du signal de sortie Y_t (fig. 11-10, *a*):

$$\delta(a_i, X_h) = a_j; \ \lambda(a_i, X_h) = Y_t$$

et soit $a_i \in A^m$, i.e. a_i est un état de l'automate S^m. Deux cas peuvent se présenter:

a) $a_j \in A^m$, i.e. a_j est un autre état de l'automate S^m. On pose alors

$$\delta^m(a_i, X_h) = \delta(a_i, X_h) = a_j; \qquad \lambda^m(a_i, X_h) = \lambda(a_i, X_h) = Y_t$$

et le sous-graphe de l'automate S^m correspondant à cette transition est le même que le sous-graphe de l'automate S de la figure 11-10, *a*.

Dans notre exemple, la troisième ligne du tableau 10-3 comporte la transition (a_1, a_6), qui se produit sous l'effet du signal $x_1x_2\bar{x}_3$, et la micro-instruction $Y_7 = \{y_2, y_{12}\}$. Comme a_1 et a_6 sont des éléments de la partie A^1 de la partition π, l'automate S^1 doit lui aussi posséder une transition (a_1, a_6):

$$\delta^1(a_1, x_1x_2\bar{x}_3) = a_6;$$

$$\lambda^1(a_1, x_1x_2\bar{x}_3) = \{y_2, y_{12}\};$$

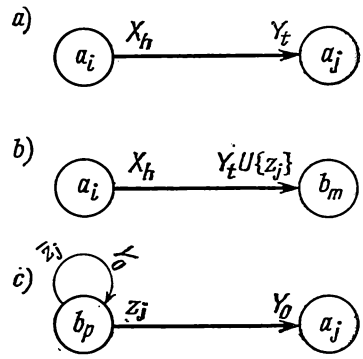


Fig. 11-10. Construction des transitions dans des automates composants

b) $a_j \in A^p$ $(p \neq m)$, i.e. a_j est un état de l'automate composant S^p. Les états a_i et a_j sont donc des états de deux automates différents. Nous dirons que dans ce cas l'automate S^m excite l'automate S^p. Définissons dans l'automate S^m les fonctions de transition et celles de sortie comme suit (fig. 11-10, *b*):

$$\delta^m(a_i, X_h) = b_m; \ \lambda^m(a_i, X_h) = Y_t \cup \{z_j\}, \qquad (11\text{-}13)$$

où z_j est une micro-opération supplémentaire élaborée dans l'automate S^m et représentant un signal d'entrée pour l'automate S^p.

Pour l'automate S^p les fonctions de transition et celles de sortie sont (fig. 11-10, *c*):

$$\begin{gathered}\delta^p(b_p, z_j) = \delta(a_i, X_h) = a_j; \quad \lambda^p(b_p, z_j) = Y_0; \\ \delta^p(b_p, \bar{z}_j) = b_p; \ \lambda^p(b_p, \bar{z}_j) = Y_0.\end{gathered} \tag{11-14}$$

Y_0 est la micro-instruction vide, le signal vecteur qui lui correspond a toutes ses composantes nulles. L'indice j de la micro-opération z_j est égal au numéro de l'état a_j de l'automate composant S^p.

Dans notre exemple, la quatrième ligne du tableau 10-3 comporte la transition (a_1, a_3), qui se produit sous l'action du signal $x_1x_2x_3$, et la micro-instruction $Y_1 = \{y_1, y_2\}$. Comme $a_1 \in A^1$ et $a_3 \in A^2$, le même signal $x_1x_2x_3$ doit produire dans l'automate S^1 une transition dans l'état supplémentaire b_1. A la sortie de S^1 apparaîtront la même micro-instruction $\{y_1, y_2\}$ et la micro-opération z_3 dont le numéro est égal à celui de l'état de transition a_3:

$$\delta^1(a_1, x_1x_2x_3) = b_1; \quad \lambda^1(a_1, x_1x_2x_3) = \{y_1, y_2, z_3\}.$$

Dans l'automate S^2 il y aura, conformément à l'expression (11-14), une transition de l'état supplémentaire b_2 dans l'état a_3 sous l'effet du signal d'entrée z_3 avec l'élaboration de la micro-instruction vide Y_0:

$$\delta^2(b_2, z_3) = a_3; \ \lambda^2(b_2, z_3) = Y_0.$$

Le nombre d'éléments de l'ensemble Z_x^m des variables d'entrée supplémentaires de l'automate composant S^m est donc égal au nombre des états, appartenant à l'ensemble $Q(A^m)$, dans lesquels dans la table des transitions de l'automate S dont on recherche une décomposition il y a des transitions à partir des états non appartenant à A^m. Le nombre d'éléments de l'ensemble Z_y^m des micro-opérations de sortie supplémentaires de l'automate composant S^m est égal au nombre d'états, appartenant à l'ensemble $T(A^m)$, dans lesquels dans la table des transitions de l'automate à décomposer S il y a des transitions à partir des états de l'ensemble A^m.

On a évidemment

$$\bigcup_{i=1}^{n} Z_x^i = \bigcup_{i=1}^{n} Z_y^i.$$

5. L'état initial a_{m1} de l'automate composant S^m sera défini comme suit:

$$a_{m1} = \begin{cases} a_1, \text{ si } a_1 \in A^m; \\ b_m \text{ dans le cas contraire.} \end{cases}$$

Ainsi donc, à l'époque initiale, tous les automates composants sont dans les états du type b_m sauf celui dont l'état initial coïncide avec l'état initial a_1 de l'automate à décomposer S.

Les tables des transitions des automates composants S^1, S^2 et S^3 sont les tableaux 11-8, 11-9, 11-10 respectivement. Le réseau C de trois automates est représenté sur la figure 11-11. La construction des tables des transitions des automates composants est immédiate. Les lignes de la table des transitions de l'automate S sont reportées, l'une après l'autre, dans les tables des transitions des automates

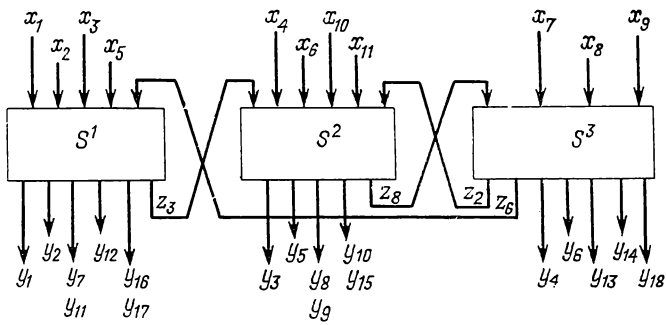


Fig. 11-11. Réseau de trois automates composants

composants. Supposons qu'au pas t on examine la ligne t de la table des transitions de l'automate S, dans laquelle est inscrite la transition $a_i a_j X_h Y_t$. Si l'état a_i appartient à l'ensemble A^m de l'automate S^m, la ligne t est reportée dans la table des transitions de l'automate S^m. Si, en outre, a_i et a_j sont deux états d'automates différentes ($a_i \in A^m$, $a_j \in A^p$), on ajoute dans la colonne Y_t de la

Tableau 11-8

Table des transitions de l'automate composant S^1

a_m	a_s	X_h	Y_t	h
a_1	a_1	$x_1\bar{x}_2$	y_1y_2	1
	a_5	$\bar{x}_1$	$y_1y_2y_{12}$	2
	a_6	$x_1x_2\bar{x}_3$	y_2y_{12}	3
	b_1	$x_1x_2x_3$	$y_1y_2z_3$	4
a_5	a_5	$\bar{x}_1$	y_7y_{11}	5
	a_6	x_1	$y_{16}y_{17}$	6
a_6	a_1	x_5	y_1y_2	7
	a_1	$\bar{x}_5$	$y_{16}y_{17}$	8
b_1	a_6	z_6	—	9
	b_1	$\bar{z}_6$	—	10

Tableau 11-9

Table des transitions de l'automate composant S^2

a_m	a_s	X_h	Y_t	h
a_2	a_2	x_6	—	1
	a_3	$\bar{x}_6$	y_3y_5	2
a_3	a_3	x_{10}	y_3y_5	3
	b_2	$\bar{x}_4\bar{x}_{10}$	$y_5y_8y_9z_8$	4
	a_9	$x_4\bar{x}_{10}$	$y_{10}y_{15}$	5
a_9	a_2	$\bar{x}_6x_{11}$	$y_5y_8y_9$	6
	a_3	$\bar{x}_{11}$	$y_3y_8y_9$	7
	a_9	x_6x_{11}	$y_{10}y_{15}$	8
b_2	a_3	z_3	—	9
	a_2	z_2	—	10
	b_2	$\bar{z}_3\bar{z}_2$	—	11

Tableau 11-10

Table des transitions de l'automate composant S^3

a_m	a_s	X_h	Y_t	h
a_4	a_4	$\bar{x}_7x_9$	$y_{13}y_{18}$	1
	b_3	x_7	$y_{13}z_6$	2
	a_8	$\bar{x}_7\bar{x}_9$	$y_{13}y_{14}$	3
a_7	b_3	x_8	$y_{14}y_{18}z_2$	4
	a_4	$\bar{x}_8$	$y_{13}y_{18}$	5
a_8	a_4	$\bar{x}_9$	y_6	6
	a_7	x_9	y_4y_6	7
b_3	a_8	z_8	—	8
	b_3	$\bar{z}_8$	—	9

table des transitions de l'automate S^m la micro-opération z_j et l'on inscrit dans la table des transitions de l'automate S^p la transition (b_p, a_j) sous l'effet du signal z_j et la micro-instruction vide $Y_0 = \varnothing$. Le nombre de lignes h^m de la table des transitions de l'automate S^m est évidemment égal à

$$h^m = h\,(A^m) + |\,Q\,(A^m)\,| + 1, \qquad (11\text{-}15)$$

où $h(A^m)$ est le nombre total des lignes de la table des transitions de l'automate S dans lesquelles sont inscrites des transitions à partir des états de l'ensemble A^m, et l'unité tient compte de la transition $b_p \rightarrow b_p$.

Il est clair, de la méthode même de décomposition, qu'à chaque époque au plus un automate peut être excité, i.e. le fonctionnement des automates composants est séquentiel, sauf à des instants d'excitation (les transitions de la figure 11-10, *b* et *c*). Alors, si l'automate S^m est excité (se trouve dans un état du type a_i), tous les autres automates composants sont dans des états du type b_p ($p = 1, \ldots, m-1, m+1, \ldots, n$). Par conséquent, on a toujours

$$A_i A_j = 0$$

pour $a_i \in A^m$ et $a_j \in A^p$. A_i et A_j sont comme auparavant les conjonctions des éléments de mémoire qui correspondent aux états a_i et a_j.

Lorsque le nombre des états $Q(A^m)$ de l'ensemble A^m dans lesquels il y a des transitions, dans l'automate à décomposer, à partir des états appartenant à d'autres ensembles, est supérieur à deux ($|Q(A^m)| > 2$), il est bon de coder les signaux d'entrée qui provoquent les transitions à partir de l'état b_m dans les états appartenant à l'ensemble $Q(A^m)$. Soit, par exemple, $Q(A^m) = \{a_1, \ldots, a_R\}$. Introduisons l'ensemble $U^m = \{u_1, \ldots, u_F\}$ de variables d'entrée binaires, où $F =]\log_2(R+1)[$. Codons chaque état a_r de l'ensemble $Q(A^m)$ par un code binaire de longueur F:

$$K(a_r) = (e_{r1}, \ldots, e_{rF}); \; e_{rf} \in \{0, 1\}, \; f = 1, \ldots, F.$$

Si alors, dans l'automate à décomposer S, il y a une transition de l'état $a_i \in A^p$ dans l'état $a_r \in A^m$ ($p \neq m$) sous l'effet du signal d'entrée X_h, il y aura dans l'automate S^m une transition de b_m dans a_r provoquée par le signal $(e_{r1}, \ldots, e_{rF})$, tandis que sa table de structure comportera au lieu du signal d'entrée z_r la conjonction

$$I_r = u_1^{e_{r1}} \ldots u_F^{e_{rF}} = \bigwedge_{f=1}^{F} u_f^{e_{rf}}; \; u_f^0 = \bar{u}_f, \; u_f^1 = u_f.$$

Le même signal, qui est cette fois un signal de sortie, sera délivré par l'automate composant S^p lorsqu'il passe de l'état a_i dans l'état b_p sous l'effet du signal d'entrée X_h. L'une des séquences d'entrée des valeurs des variables $u_1, \ldots, u_F$, par exemple la séquence nulle, est réservée au signal d'entrée qui maintient l'automate S^m dans l'état b_m lorsque l'un quelconque des autres automates composants est excité.

11-5. Synthèse des automates microprogrammés avec des MLP à mémoire

S'il s'agit de réaliser un automate microprogrammé $S = (A, X, Y, \delta, \lambda, a_1)$ dont le nombre L de variables d'entrée ou le nombre H des lignes de la table de structure dépassent respectivement le nombre d'entrées ou celui de termes de MLP (s, t, q, r), on peut procéder par les méthodes de décomposition décrites plus haut. Le problème consiste en l'occurrence de rechercher, d'après une partition choisie $\pi = \{A^1, \ldots, A^n\}$ sur l'ensemble A des états de l'automate S, un réseau d'automates composants $S^m = (B^m, X^m, Y^m, \delta^m, \lambda^m, a_{m1})$, $m = 1, \ldots, n$, qu'on réalise ensuite avec des MLP (s, t, q, r). Les conditions suivantes sont à respecter:

1) le nombre de variables d'entrée à appliquer sur chacun des automates S^m, y compris les variables d'entrée supplémentaires en provenance d'autres automates composants, ne dépasse pas le nombre s d'entrées de la MLP (s, t, q, r);

2) le nombre de variables de sortie (micro-opérations) de chacun des automates S^m, y compris les variables de sortie supplémentaires prévues pour l'excitation d'autres automates composants, ne dépasse pas le nombre t de sorties de la MLP (s, t, q, r);

3) le nombre h^m de lignes de la table de structure de chaque automate S^m ne dépasse pas le nombre de termes de la MLP (s, t, q, r);

4) le nombre R^m d'éléments de mémoire nécessités par chaque automate S^m ne dépasse pas le nombre r d'éléments de mémoire dont dispose une MLP (s, t, q, r).

Ainsi donc, le problème de la synthèse d'un automate microprogrammé S avec des MLP (s, t, q, r) se ramène à la recherche d'une partition $\pi = \{A^1, \ldots, A^n\}$ à nombre minimal de parties, dont chaque ensemble A^m définit un automate composant S^m $(m = 1, \ldots, n)$ tel que:

$$|X^m| \leqslant s; \tag{11-16}$$

$$|Y^m| \leqslant t; \tag{11-17}$$

$$h^m \leqslant q; \tag{11-18}$$

$$R^m \leqslant r. \tag{11-19}$$

Ce problème diffère essentiellement du problème, formulé au § 11-2, de la synthèse d'un AMP S avec des MLP combinationnelles par la nécessité de tenir compte de l'ensemble des liaisons Z_x^m et Z_y^m $(m = 1, \ldots, n)$ entre les automates composants du réseau C réalisant l'automate initial S.

Voyons comment se cherche la partition π par un algorithme approché séquentiel. L'exposé sera illustré sur l'exemple de la synthèse de l'automate donné par le tableau 10-3, qu'on se propose de réaliser avec des MLP (10, 10, 20, 5) ayant 10 entrées, 10 sor-

ties, 20 termes et 5 bascules du type D. D'après le tableau 10-3, on construit le tableau 11-11, en mettant en correspondance à chaque état a_i les ensembles $X(a_i)$ et $Y(a_i)$ des variables d'entrée et des variables de sortie inscrites en regard de toutes les transitions à partir de a_i; l'ensemble $G(a_i)$ des états à partir desquels il y a des transitions dans a_i ($a_i \notin G(a_i)$); l'ensemble $T(a_i)$ des états dans

Tableau 11-11

Caractéristiques des transitions à partir des états de l'AMP S

a_i	$X(a_i)$	$Y(a_i)$	$G(a_i)$	$T(a_i)$	$h(a_i)$
a_1	x_1, x_2, x_3	$y_1, y_2 y_{12}$	a_6	a_3, a_5, a_6	4
a_2	x_6	y_3, y_5	a_7, a_9	a_3	2
a_3	x_4, x_{10}	y_3, y_5, y_8, y_{10}	a_1, a_2, a_9	a_8, a_9	3
a_4	x_7, x_9	y_{13}, y_{14}, y_{18}	a_7, a_8	a_6, a_8	3
a_5	x_1	y_7, y_{16}	a_1	a_6	2
a_6	x_5	y_1, y_2, y_{16}	a_1, a_4, a_5	a_1	2
a_7	x_8	y_{13}, y_{14}, y_{18}	a_8	a_2, a_4	2
a_8	x_9	y_4, y_6	a_3, a_4	a_4, a_7	2
a_9	x_6, x_{11}	y_3, y_5, y_8, y_{10}	a_3	a_2, a_3	3

lesquels il y a des transitions de l'état a_i, ainsi que le nombre $h(a_i)$ de lignes qui comportent des transitions à partir de l'état a_i. De même que précédemment, on a omis d'inscrire dans le tableau 11-11 les micro-opérations y_{11}, y_9, y_{15}, y_{17} qui se rencontrent, dans le tableau 10-3, toujours en association avec respectivement les micro-opérations y_7, y_8, y_{10} et y_{16}. On voit immédiatement du tableau 11-11 qu'on ne peut pas combiner arbitrairement les états de l'automate initial en vue d'en former des ensembles de la partition cherchée π, car il peut s'avérer que l'automate composant correspondant ne saurait être réalisé avec une seule MLP (10, 10, 20, 5). Par exemple, si $A^m = \{a_1, a_2, a_3, a_4\}$, deux contraintes, (11-16) et (11-17), se trouvent violées à la fois, car pour l'automate S^m on aurait: $X(A^m) = \{x_1, x_2, x_3, x_4, x_6, x_7, x_9, x_{10}\}$; $Z_x^m = \{z_1, z_2, z_3, z_4\}$, puisque dans chacun des états a_1, a_2, a_3, a_4 il y a des transitions à partir des états qu'on n'a pas inclus dans A^m (cf. les lignes 5 à 9 dans la colonne $T(a_i)$ du tableau 11-11); $X^m = X(A^m) \cup Z_x^m$, $|X^m| = 12$; $Y(A^m) = \{y_1, y_2, y_3, y_5, y_8, y_{10}, y_{12}, y_{13}, y_{14}, y_{18}\}$; $Z_y^m = \{z_5, z_6, z_8, z_9\}$, puisque $a_5, a_6, a_8, a_9 \in T(A^m)$ (cf. les lignes 1 à 4 dans la colonne $T(a_i)$ du tableau 11-11); $Y^m = Y(A^m) \cup Z_y^m$, $|Y^m| = 14$.

Tout comme dans le § 11-2, les trois critères essentiels suivants seront utilisés dans l'algorithme séquentiel :

1. $c(a_i, a_j)$, proximité de deux états a_i et a_j ;
2. $c(a_i, A^m)$, proximité de l'état a_i et de l'ensemble A^m ;
3. $w(a_i, A^m)$, fonction de poids de l'état a_i relativement à l'ensemble $A^m \subset A$. Ici a_i est l'état non inclus dans l'ensemble A^m ($a_i \in A \setminus A^m$), et $w(a_i, A^m)$ est le critère de choix de l'état suivant à inclure dans l'ensemble A^m de la partition π.

La proximité de deux états se définit par l'expression

$$c(a_i, a_j) = k_1 \cdot |X(a_i) \cap X(a_j)| + k_2 \cdot |Y(a_i) \cap Y(a_j)| + \\ + k_3 (|G(a_i) \cap G(a_j)| + |T(a_i) \cap T(a_j)| + |\{a_i\} \cap T(a_j)| + \\ + |\{a_j\} \cap T(a_i)|). \qquad (11\text{-}20)$$

Le terme de (11-20) affecté du coefficient k_3 permet d'estimer le nombre d'états communs à partir desquels il y a des transitions dans a_i et a_j ($G(a_i) \cap G(a_j)$), ceux dans lesquels il y a des transitions de a_i et a_j ($T(a_i) \cap T(a_j)$), ainsi que de juger si les états a_i et a_j eux-mêmes font partie des ensembles d'états dans lesquels il y a des transitions de a_j et a_i respectivement. Dans notre exemple, posons $k_1 = k_2 = k_3 = 1$. On a $c(a_2, a_3) = |\{x_6\} \cap \{x_4, x_{10}\}| +$

$$+ |\{y_3, y_5\} \cap \{y_3, y_5, y_8, y_{10}\}| + |\{a_7, a_9\} \cap \{a_1, a_2, a_9\}| + \\ + |\{a_3\} \cap \{a_8, a_9\}| + |\{a_2\} \cap \{a_8, a_9\}| + |\{a_3\} \cap \{a_3\}| = 4.$$

La proximité de l'état a_i et de l'ensemble A^m sera définie comme suit :

$$c(a_i, A^m) = \sum_{a_j \in A^m} c(a_i, a_j). \qquad (11\text{-}21)$$

La fonction de poids $W(a_i, A^m)$ peut être définie par la relation

$$W(a_i, A^m) = c(a_i, A^m) - c(a_i, A \setminus A^m), \qquad (11\text{-}22)$$

c'est la différence des proximités entre l'état a_i et l'ensemble A^m d'une part et l'état a_i et l'ensemble $A \setminus A^m$ des états non appartenant à A^m d'autre part.

L'algorithme de recherche de la partition π est presque le même que celui que nous avons employé pour la recherche de la partition π_E lors de la synthèse d'un AMP avec des MLP combinationnelles. La formation de l'ensemble A^m se fait en deux étapes. A la première étape, on inclut dans l'ensemble A^m l'élément de base — l'état a_i — tel que

$$c(a_i, A' \setminus \{a_i\}) = \max_j c(a_j, A' \setminus \{a_j\}),$$

où $A' = A \setminus \left(\bigcup_{k=1}^{m-1} A^k\right)$ est l'ensemble des états non rapportés, vers l'instant considéré, à aucune partie ; $a_i, a_j \in A'$, et $c(a_j, A' \setminus \{a_j\})$

est défini par la formule (11-21). Ainsi donc, en tant qu'élément de base de l'ensemble A^m on choisit l'état le plus proche de tous les états qui attendent à être répartis, i.e. aux états non inclus dans les ensembles $A^1, \ldots, A^{m-1}$.

A la deuxième étape, itérée, on calcule à chaque pas, par la formule (11-22), la fonction de poids $w(a_t, A^m)$ pour chacun des états $a_t \in A \setminus \left(\bigcap_{k=1}^{m} A^k\right)$ tels que $c(a_t, A^m) \neq 0$. Parmi les états vérifiant cette condition on choisit un état a_j avec $w(a_j, A^m) = \max_t w(a_t, A^m)$ si son inclusion dans l'ensemble A^m n'enfreint aucune des restrictions (11-16) à (11-19). Lorsque ce critère est le même pour plusieurs états, on choisit parmi eux celui qui présente $\max_t c(a_t, A^m)$; dans le cas où ce critère est lui aussi égal pour plusieurs états, on inclut dans l'ensemble A^m l'un quelconque de ces états.

Si les restrictions (11-16) à (11-19) s'opposent à l'inclusion dans l'ensemble A^m de l'état a_j présentant la fonction de poids maximale, on examine, dans l'ordre de décroissance de leur fonction de poids, les autres états non répartis dont la proximité de l'ensemble A^m n'est pas nulle. On en choisit le premier venu conforme aux conditions (11-16) à (11-19). En dernier lieu et dans le même ordre on examine les états a_t avec $c(a_t, A^m) = 0$. Le processus de formation de l'ensemble A^m est terminé si la liste des états est épuisée ou bien si les restrictions (11-16) à (11-19) n'autorisent d'y inclure aucun des états qui restent.

Reprenons notre exemple de synthèse de l'AMP donné par le tableau 10-3 avec des MLP (10, 10, 20, 5). Les données initiales pour le calcul des critères $c(a_i, a_j)$, $c(a_i, A^m)$ et $w(a_i, A^m)$ sont

Tableau 11-12

Matrice de proximité des états

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9
a_1	—	1	1	1	3	4	0	0	1
a_2	1	—	4	1	0	0	1	0	5
a_3	1	4	—	1	1	1	0	1	6
a_4	1	1	1	—	1	1	5	3	0
a_5	3	0	1	1	—	3	0	0	0
a_6	4	0	1	1	3	—	0	1	0
a_7	0	1	0	5	0	0	—	2	1
a_8	0	0	1	3	0	1	2	—	1
a_9	1	5	6	0	0	0	1	1	—

Tableau 11-13

Formation des ensembles de la partition π

A	$c(a_i, A^1)$	$w(a_i, A^1)$							$c(a_i, A^1)$	$w(a_i, A^2)$		
		1	2	3	4	5	6	7		8	9	10
a_1	11	—9	—7	—5	—5	—3	—3	—	7	⊕	+	+
a_2	12	—4	6	⊕	+	+	+	+				
a_3	15	⊕	+	+	+	+	+	+				
a_4	13	—11	—13	—11	—1	⊕	+	+				
a_5	8	—6	—6	—6	—6	—4	—4	—	6	0	6	⊕
a_6	10	—8	—8	—8	—6	—2	—2	—	7	1	⊕	+
a_7	9	—9	—7	—5	—1	11	⊕	+				
a_8	8	—6	—4	—4	⊕	+	+	+				
a_9	14	—2	⊕	+	+	+	+	+				

réunies dans le tableau 11-11. Le tableau 11-12 donne la matrice de proximité des états avec $k_1 = k_2 = k_3 = 1$, et le tableau 11-13, analogue par sa structure au tableau 11-3, le processus de formation des ensembles de la partition π. Dans l'exemple considéré, la partition π se compose de deux ensembles:

$$\pi = \{A^1, A^2\};$$

$$A^1 = \{a_2, a_3, a_4, a_7, a_8, a_9\}, \; A^2 = \{a_1, a_5, a_6\}.$$

Les tableaux 11-14 et 11-15 sont les tables de structure des automates construits en conformité de la partition π, la figure 11-12

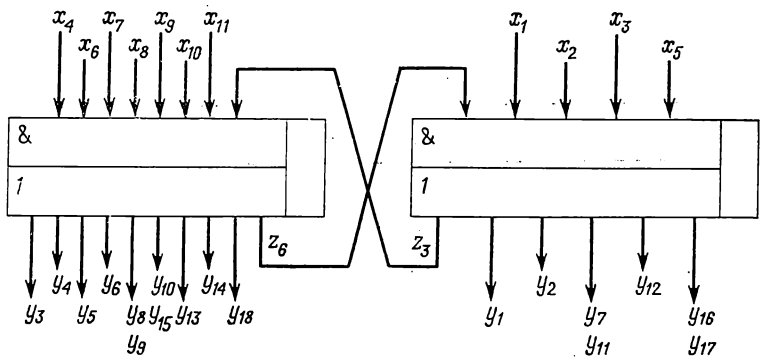


Fig. 11-12. Schéma logique d'un AMP avec des MLP (10, 10, 20, 5)

en donne le schéma logique. Les tableaux 11-16 et 11-17 illustrent la programmation des MLP en conformité des tables de structure

Tableau 11-14

Automate composant S^1

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t	D_h	h
a_2	000	a_2	000	x_6	—	—	1
		a_3	001	$\bar{x}_6$	y_3y_5	D_3	2
a_3	001	a_3	001	x_{10}	y_3y_5	D_3	3
		a_8	100	$\bar{x}_4\bar{x}_{10}$	y_5y_8	D_1	4
		a_9	101	$x_4\bar{x}_{10}$	y_{10}	D_1D_3	5
a_4	010	a_4	010	$\bar{x}_7x_9$	$y_{13}y_{18}$	D_2	6
		b_1	110	x_7	$y_{13}z_6$	D_1D_2	7
		a_8	100	$\bar{x}_7\bar{x}_9$	$y_{13}y_{14}$	D_1	8
a_7	011	a_2	000	x_8	$y_{14}y_{18}$	—	9
		a_4	010	$\bar{x}_8$	$y_{13}y_{18}$	D_2	10
a_8	100	a_4	010	$\bar{x}_9$	y_6	D_2	11
		a_7	011	x_9	y_4y_6	D_2D_3	12
a_9	101	a_2	000	$\bar{x}_6x_{11}$	y_5y_8	—	13
		a_3	001	$\bar{x}_{11}$	y_3y_8	D_3	14
		a_9	101	x_6x_{11}	y_{10}	D_1D_3	15
b_1	110	a_3	001	z_3	—	D_3	16
		b_1	110	$\bar{z}_3$	—	D_1D_2	17

Tableau 11-15

Automate composant S^2

a_m	$K(a_m)$	a_s	$K(a_s)$	X_h	Y_t	D_h	h
a_1	00	a_1	00	$x_1\bar{x}_2$	y_1y_2	—	1
		a_5	01	$\bar{x}_1$	$y_1y_2y_{12}$	D_2	2
		a_6	10	$x_1x_2\bar{x}_3$	y_2y_{12}	D_1	3
		b_2	11	$x_1x_2x_3$	$y_1y_2z_3$	D_1D_2	4
a_5	01	a_5	01	$\bar{x}_1$	y_7	D_2	5
		a_6	10	x_1	y_{16}	D_1	6
a_6	10	a_1	00	x_5	y_1y_2	—	7
		a_1	00	$\bar{x}_5$	y_{16}	—	8
b_2	11	a_6	10	z_6	—	D_1	9
		b_2	11	$\bar{z}_6$	—	D_1D_2	10

Tableau 11-16

Programmation de la MLP_1 (10, 10, 20, 5)

nos	Entrées										Mémoire					Termes	Sorties										Excitation de la mémoire				
	x_4	x_6	x_7	x_8	x_9	x_{10}	x_{11}	z_3	—	—	T_1	T_2	T_3	T_4	T_5		y_3	y_4	y_5	y_6	y_8	y_{10}	y_{13}	y_{14}	y_{18}	z_6	D_1	D_2	D_3	D_4	D_5
	1	2	3	4	5	6	7	8	9	10							1	2	3	4	5	6	7	8	9	10					
1	—	0	—	—	—	—	—	—	—	—	0	0	0	—	—	e_2	1	·	1	·	·	·	·	·	·	·	·	·	1	·	·
2	—	—	—	—	—	1	—	—	—	—	0	0	1	—	—	e_3	1		1		·						·		1		
3	0	—	—	—	—	0	—	—	—	—	0	0	1	—	—	e_4	·		1		1	·					1	·	·		
4	1	—	—	—	—	0	—	—	—	—	0	0	1	—	—	e_5			·			1	·		·		1	·	1	·	
5	—	—	0	—	1	—	—	—	—	—	0	1	0	—	—	e_6			·				1		1	·	·	1	·		
6	—	—	1	—	—	—	—	—	—	—	0	1	0	—	—	e_7						·	1	·	·	1	1	1			
7	—	—	0	—	0	—	—	—	—	—	0	1	0	—	—	e_8						·	1	1	·		1	·			
8	—	—	—	1	—	—	—	—	—	—	0	1	1	—	—	e_9							·	1	1	·		·			
9	—	—	—	0	—	—	—	—	—	—	0	1	1	—	—	e_{10}				·			1	·	1		·	1			
10	—	—	—	—	0	—	—	—	—	—	1	0	0	—	—	e_{11}		·		1					·		·	1	·		
11	—	—	—	—	1	—	—	—	—	—	1	0	0	—	—	e_{12}		1	·	1	·			·				1	1	·	
12	—	0	—	—	—	—	1	—	—	—	1	0	1	—	—	e_{13}	·	·	1		1		·	·				·	·		
13	—	—	—	—	—	—	0	—	—	—	1	0	1	—	—	e_{14}	1		·	·	1	·	·				·		1		
14	—	1	—	—	—	—	1	—	—	—	1	0	1	—	—	e_{15}	·				·	1					1		1	·	
15	—	—	—	—	—	—	—	1	—	—	1	1	0	—	—	e_{16}						·					·	·	1	·	
16	—	—	—	—	—	—	—	0	—	—	1	1	0	—	—	e_{17}											1	1	·	·	·
17	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—											·	·			
...																															
20	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

Tableau 11-17

Programmation de la MLP_2 (10, 10, 20, 5)

nos	Entrées										Mémoire					Termes	Sorties										Excitation de la mémoire				
	x_1	x_2	x_3	x_5	z_6	—	—	—	—	—	T_1	T_2	T_3	T_4	T_5		y_1	y_2	y_7	y_{12}	y_{16}	z_3	—	—	—	—	D_1	D_2	D_3	D_4	D_5
	1	2	3	4	5	6	7	8	9	10							1	2	3	4	5	6	7	8	9	10					
1	1	0	—	—	—	—	—	—	—	—	0	0	—	—	—	e_1	1	1					·	·	·	·	·				·
2	0	—	—	—	—	—	—	—	—	—	0	0	—	—	—	e_2	1	1		1			·	·	·	·	·	1			·
3	1	1	0	—	—	—	—	—	—	—	0	0	—	—	—	e_3	·	1		1			·	·	·	·	1				·
4	1	1	1	—	—	—	—	—	—	—	0	0	—	—	—	e_4	1	1				1	·	·	·	·	1	1			·
5	0	—	—	—	—	—	—	—	—	—	0	1	—	—	—	e_5	·	·	1				·	·	·	·	·	1		·	·
6	1	—	—	—	—	—	—	—	—	—	0	1	—	—	—	e_6	·	·			1		·	·	·	·	1				·
7	—	—	—	1	—	—	—	—	—	—	1	0	—	—	—	e_7	1	1						·	·	·	·				·
8	—	—	—	0	—	—	—	—	—	—	1	0	—	—	—	e_8	·				1		·	·	·	·	·			·	·
9	—	—	—	—	1	—	—	—	—	—	1	1	—	—	—	e_9	·				·		·	·	·	·	1				·
10	—	—	—	—	0	—	—	—	—	—	1	1	—	—	—	e_{10}	·						·	·	·	·	1	1			·
11	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	·								·	·	·	·			·
...																															
20	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	·						·	·	·	·	·				·

des automates (tableaux 11-14 et 11-15). On remarque que le tableau 11-16 ne comporte pas de terme e_1 correspondant à la première ligne du tableau 11-14. La raison en est qu'il n'entre dans aucune fonction d'excitation de mémoire ou fonction de sortie. Comparant les figures 11-12 et 11-8 on remarque qu'avec des MLP à mémoire on a pu résoudre le problème avec un nombre de MLP moindre que dans le cas des MLP combinationnelles, pour des paramètres s, t et q identiques.

La recherche de la partition π sur l'ensemble A des états de l'AMP S, dans le cas où l'on se propose de le réaliser avec des MLP (s, t, q, r), se trouve notablement compliquée par les restrictions (11-16) et (11-17), car même dans les automates de complexité moyenne le nombre L de variables d'entrée et le nombre N de variables de sortie dépassent de beaucoup les nombres s et t d'entrées et de sorties respectivement d'une MLP (s, t, q, r). Il est donc désirable que deux automates composants quelconques S^m et S^p aient l'intersection de leurs ensembles de variables d'entrée $X(A^m)$ et $X(A^p)$ ou de sortie $Y(A^m)$ et $Y(A^p)$ vide ou, à la rigueur, minimale (ou des deux à la fois). On y parvient, pour autant qu'il est possible, en introduisant certaines relations sur l'ensemble A des états de l'AMP à l'appui de sa table des transitions.

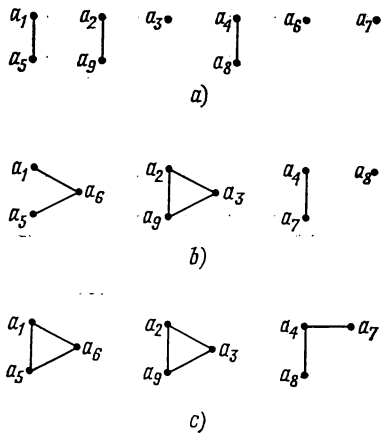


Fig. 11-13. Graphes des relations ω (a), φ (b) et α (c)

Ainsi, au paragraphe 10-3, en procédant au changement de variables au cours de la réalisation matricielle d'un AMP, nous avons introduit une relation ω. Rappelons que deux états a_i et a_j ($i \neq j$) d'un automate S vérifient la relation ω ($a_i \omega a_j$) si et seulement si l'intersection des ensembles $X(a_i)$ et $X(a_j)$ n'est pas vide : $X(a_i) \cap X(a_j) \neq \varnothing$. Construisons le graphe Γ_ω de cette relation (fig. 11-13, a) pour l'automate S donné par la table 10-3. Au graphe Γ_ω on peut mettre en correspondance, sur l'ensemble A, une partition π_ω de façon à associer dans un sous-ensemble de π_ω les états d'une même composante connexe de Γ_ω. La situation illustrée par la figure 11-13, a est la suivante :

$$\pi = \overline{1.5}, \overline{2.9}, \overline{3}, \overline{4.8}, \overline{6}, \overline{7}.$$

Soient A^m et A^p les ensembles de sommets des composantes connexes m-ième et p-ième respectivement du graphe Γ_ω. Si

$X(A^m) = \bigcup_{a_i \in A^m} X(a_i)$ et $X(A^p) = \bigcup_{a_j \in A^p} X(a_j)$, on a

$$X(A^m) \cap X(A^p) = \varnothing \tag{11-23}$$

pour $m \neq p$, $m, p = 1, \ldots, n$ quelconques, n étant le nombre de composantes connexes du graphe Γ_ω. Si alors, dans le problème de décomposition de l'automate S, on choisit π_ω en tant que partition initiale, pour tout couple d'automates composants l'intersection de leurs ensembles de variables d'entrée sera vide.

Souvent le réseau de décomposition cherché doit vérifier, outre les restrictions exprimées par la relation (11-23), certaines autres restrictions. On peut, par exemple, donner à l'avance le nombre d'automates composants qui peut s'avérer plus petit que celui des ensembles de la partition π_ω, ou bien le nombre d'états dans les automates composants doit être plus grand que le nombre d'éléments de la partie correspondante de la partition π_ω. On construit alors, au départ de la partition π_ω, une partition $\pi'_\omega > \pi_\omega$ en réunissant certaines parties de π_ω en une seule. Lorsque, dans le problème de décomposition, on choisit en tant que partition initiale la partition π'_ω, elle aussi vérifiera les relations du type (11-23).

De même, on peut donner sur l'ensemble des états de l'automate S une relation φ telle que $a_i \varphi a_j$ $(i \neq j)$ si et seulement si l'intersection des ensembles respectifs de micro-opérations dans toutes les transitions à partir des états a_i et a_j n'est pas vide : $Y(a_i) \cap \cap Y(a_j) \neq \varnothing$. Alors, si l'on choisit en tant que partition initiale la partition π_φ ou une partition $\pi'_\varphi > \pi_\varphi$, pour tout couple d'automates composants S^m et S^p les ensembles de variables de sortie n'ont pas eux non plus de variables communes : $Y(A^m) \cap Y(A^p) = \varnothing$.

Le graphe Γ_φ de l'automate S (fig. 11-13, *b*) nous donne

$$\pi_\varphi = \overline{1.5.6},\ \overline{2.3.9},\ \overline{4.7},\ \overline{8}.$$

Si l'on définit, sur l'ensemble A, une relation α telle que $a_i \alpha a_j$ $(i \neq j)$ si et seulement si l'intersection des ensembles des variables d'entrée ou des micro-opérations pour toutes les transitions à partir des états a_i et a_j n'est pas vide, i.e. $(X(a_i) \cap X(a_j) \neq \varnothing) \vee \vee (Y(a_i) \cap Y(a_j) \neq \varnothing)$, alors en admettant comme partition initiale π_α ou une partition $\pi'_\alpha > \pi_\alpha$, on s'assure la séparation des variables d'entrée et de sortie des automates composants. Directement de la figure 11-13, *c* on obtient pour l'automate S la partition $\pi_\alpha = \overline{1.5.6},\ \overline{2.3.9},\ \overline{4.7.8}$, laquelle a été utilisée dans le § 11-4 pour la réalisation de l'AMP S à l'aide d'un réseau de trois automates composants. La solution la plus intéressante, du point de vue pratique, du problème de la décomposition est celle où les variables d'entrée ou les variables de sortie (ou les unes comme les autres)

des automates composants se séparent, et en outre le nombre de liaisons entre eux est minimal. Le problème se ramène à la recherche d'une partition $\pi'_\omega \geqslant \pi_\omega$ ($\pi'_\varphi \geqslant \pi_\varphi$ ou $\pi'_\alpha \geqslant \pi_\alpha$), plus exactement, on cherche à réunir entre eux certains ensembles de la partition π_ω (resp. π_φ ou π_α) de façon à minimiser le nombre de liaisons entre les états échus dans différents ensembles. Ainsi, par exemple, la partition $\pi = \overline{1.5.6}, \overline{2.3.4.7.8.9}$ utilisée lors de la synthèse de l'automate S avec des MLP (10, 10, 20, 5) s'obtient par fusionnement du deuxième et du troisième ensemble de la partition π_α ($\pi > \pi_\alpha$). Remarquons pour conclure qu'il existe des automates qui n'admettent pas de partitions sur les ensembles de variables d'entrée ou de sortie, tandis que l'algorithme séquentiel que nous venons de considérer permet toujours de minimiser le nombre d'éléments dans l'intersection des ensembles d'entrée et de sortie.

BIBLIOGRAPHIE *)

1*. ATSTOPAS F. F., GINTELIS G. B., CHOMSKIS R. R. Langage d'entrée pour une synthèse structurale automatique d'ordinateurs. *In*: Comptes rendus de la XXe conférence scientifico-technique de la Lituanie. Kaunas, 1970, t. 2, Calcul électronique.

2. AUFENKAMP D. D., HOHN F. E. Analysis of sequential machines. *IRE Trans.*, vol. EC-6, n° 4, 1957.

3*. BARACHENKOV V. V., PAPKOV V. I., STÉPACHKINE G. I. Les calculateurs numériques automatiques. Ed. de LETI, 1975, fasc. 4. Systèmes de commande microprogrammée.

4*. BARANOV S. I. Minimisation des sommets conditionnels dans les schémas-graphes d'algorithmes. *In*: Travaux du symposium IFAC, Systèmes numériques. Riga, « Zinatne », 1974, t. 1.

5*. BARANOV S. I. Synthèse des automates microprogrammés. Léningrad, « Energuia », 1974.

6*. BARANOV S. I. Synthèse d'un modèle composite d'automate microprogrammé. *In*: Synthèse des automates et commande par des réseaux de télécommunications. Moscou, « Naouka », 1973.

7. BARANOV S. I. Matrix Realization of Control Automata. Proc. of the 2^{d} international symp. on discrete systems. Dresde, 1977, vol. 2.

8. BARANOV S. I., KEEVALLIK A. E. Transformation of graph-schemes of algorithms. *Digital processes*, 1980, vol. 6, n°s 2-3.

9*. BARANOV S. I., ROSSINOV V. V. Applications des graphes d'algorithmes à la synthèse d'automates numériques. *In*: Questions de la théorie des calculateurs numériques électroniques, fasc. 1, Kiev, 1968.

10. BARANOV S. I., KEEVALLIK A. E. Synthesis of control automata using graph-schemes of algorithms. *Digital processes*, 1980, vol. 6, n°s 2-3.

11*. BARANOV S. I., SINEV V. N. Automates et matrices programmables. Minsk, « Vyschaïa schkola », 1980.

12*. BARANOV S. I., KILLENBERG Kh. Décomposition d'automates microprogrammés. *Avtomatika i vytchislit. tekhnika*, 1978, n° 6.

13. CIOFFI G., CONSTANTINI E., JULIS S. A new approach to the decomposition of sequential systems. *Digital processes*, 1977, n° 3.

14. CURTIS H. A. Multiple Reduction of Variable Dependency of Sequential Machines. *Journal of the Association for Computing Machinery*, 1962, vol. 9, n° 3.

15*. DIATCHENKO V. F., LASAREV V. G., SAVVINE G. G. Commande par des réseaux de télécommunications. Moscou, « Naouka », 1967.

16*. ERCHOV A. P. Algorithmes opératoriels (sur les schémas opératoriels de Yanov). Problèmes de cybernétique, 1968, fasc. 20.

17*. FRIZNOVITCH G. F. Une approche de l'automatisation de la synthèse des automates finis asynchrones. *In*: Tanul. MTA Számitástechn. és Automat. Kutató Intézet. Budapest, 21, 1974.

18*. FRIZNOVITCH G. F. Synthèse de décodeurs de micro-instructions avec des MLP. *Avtomatika i vytchislit. tekhnika*, 1981, n° 2.

19*. GAVRILOV M. A., DÉVIATKOV V. V., POUPYREV E. I. Etudes logiques d'automates séquentiels. Moscou, « Naouka », 1977.

20*. GAVRILOV M. A., DÉVIATKOV V. V., TCHITCHKOWSKI A. B. Le langage des schémas opératoriels d'algorithmes parallèles avec mémoire. *In*: La théorie structurale et abstraite des réseaux de commutation à relais (automatisation de la synthèse logique). Moscou, « Naouka », 1975.

21*. GLOUCHKOV V. M. Sur l'application de la théorie abstraite des automates à la minimisation des microprogrammes. *Izvestia* de l'Académie des Sciences de l'U.R.S.S. Cybernétique technique, 1964, n° 1.

*) Les références marquées d'un astérisque ne sont disponibles qu'en russe.

22*. GLOUCHKOV V. M. Synthèse des automates numériques. Moscou, « Fizmatguiz », 1962.
23*. GORBATOV V. A. Théorie des systèmes partiellement ordonnés. Moscou, « Sovetskoe radio », 1976.
24. GORMAN K. The Programmable Logic Array: a New Approach to Microprogramming. *Electronic Design News*, 1973, vol. 18, n° 22.
25. GRASSELLI A., LUCCIO F. A Method for Minimizing the Number of Internal States in Incompletely Specified Sequential Networks. *IEEE Transactions on Electronic Computers*, 1965, vol. EC-14, n° 3.
26. HARTMANIS J. Loop Free Structure of Sequential Machines. *Information and Control*, 1962, vol. 5, n° 1.
27. HARTMANIS J. On the State Assignment Problem for Sequential Machines. *IRE Transactions on Electronic Computers*, 1961, vol. EC-10, n° 2.
28. HARTMANIS J., STEARNS R. Algebraic Structure Theory of Sequential Machines. New York, Prentice-Hall, 1966.
29*. HURWITZ E. A. Synthèse des machines séquentielles polysynchrones. Moscou, « Sviaz », 1969.
30. JAKOBSON G., KEEVALLIK A., LEIS P. Some aspects of the construction of microprogram automata networks. *In*: Proc. of the 2d International Symp. on discrete systems. Dresde, 1977, vol. 1.
31*. KALOUZHNINE L. A. Introduction à l'algèbre générale. Moscou, « Fizmatguiz », 1973.
32*. KALOUZHNINE L. A. Sur l'algorithmisation des problèmes mathématiques. *Problemy kibernetiki*, 1959, fasc. 2.
33*. KAPITONOVA J. V. Codage des automates abstraits par des codes adjacents. *Kibernetika*, 1965, n° 1.
34. KAUFMANN A. Introduction à la combinatorique en vue des applications. Paris, Dunod, 1968.
35*. KEEVALLIK A. E. Théorème de décomposition des automates finis. *Avtomatika i vytchislit. tekhnika*, 1974, n° 1.
36. KELLA J. State Minimization of Incompletely Specified Sequential Machines. *IEEE Transactions on Computers*, 1970, vol. C-19, n° 4.
37. KOHAVI ZVI. Secondary State Assignment for Sequential Machines. *IEEE Transactions on Electronic Computers*, 1964, vol. EC-13, n° 3.
38*. LASAREV V. G., PIJL E. I. Synthèse des automates de commande. Moscou, « Energuia », 1978.
39*. LIAPOUNOV A. A. Sur les schémas logiques des programmes. *Problemy kibernetiki*, 1958, fasc. 1.
40*. MAJOROV S. A., NOVIKOV G. I. Principes d'organisation de calculateurs numériques. Léningrad, « Machinostrojenie », 1974.
41*. MATSEVITY L. V. Méthodes de synthèse d'ensembles de calculateurs numériques avec des éléments du type potentiel. Kiev, 1963.
42*. MATSEVITY L. V., DÉNISSENKO E. L. Sur le codage des états internes de certains types de dispositifs multitemps. *Kibernetika*, 1966, n° 1.
43. MEISEL W. S. A Note on Internal State Minimization in Incompletely Specified Sequential Networks. *IEEE Transactions on Electronic Computers*, 1967, vol. EC-16, n° 4.
44*. MÉLIKHOV A. N. Graphes orientés et automates finis. Moscou, « Naouka », 1971.
45. MILLER R. E. Switching theory. Vol. 1: Combinational circuits. N.Y., Wiley, 1965.
46. MILLER R. E. Switching theory. Vol. 2: Sequential circuits. N.Y., Wiley, 1965.
47*. MICHTCHENKO A. T. Sur la spécification des automates par un microprogramme. *Kibernetika*, 1970, n° 3.

48*. MOROZ D. Z., MOROZOV B. V. Etudes expérimentales de quelques algorithmes de codage des états. *In*: Théorie des automates finis et ses applications. Riga, « Zinatne », 1974, fasc. 3.
49*. NOVIKOV P. S. Eléments de logique mathématique. Moscou, « Fizmatguiz », 1960.
50. PAULL M. C., UNGER S. H. Minimizing the Number of States in Incompletely Specified Sequential Switching Functions. *IRE Transactions on Electronic Computers*, 1959, vol. EC-8, n° 3.
51*. POTTOSSINE You. V. Méthodes de minimisation du nombre d'états de l'automate numérique (tour d'horizon), *Avtomatika i vytchislit. tekhnika*, 1971, n° 8.
52*. POTTOSSINE You. V. Deux algorithmes de minimisation du nombre d'états de l'automate numérique. *Avtomatika i vytchislit. tekhnika*, 1967, n° 4.
53. QUINE W. V. The Problem of Simplifying of Truth Functions. *Amer. Math. Monthly*, 1952, vol. 59, n° 8.
54*. ROUVINSKI B. I. Réalisation d'automates microprogrammés en IGE. *Elektronika. Microelektronika*, 1974, fasc. 5 (53).
55*. SCHIKHANOVITCH You. A. Introduction aux mathématiques contemporaines. Moscou, « Fizmatguiz », 1965.
56*. SCHREIDER You. A. Egalité, congruence, ordre. Moscou, « Fizmatguiz », 1971.
57*. SIDOROV You. N. Automatisation de la synthèse d'automates de commande incomplètement spécifiés. *In*: Théorie des circuits à relais. Tchéliabinsk, 1976.
58. STEARNS R. E., HARTMANIS J. On the State Assignment Problem for Sequential Machines, II. *IRE Transactions on Electronic Computers*, 1961, vol. EC-10, n° 4.
59*. TCHÉBOTAREV A. N. Synthèse abstraite d'automates de commande d'après un microprogramme. *Kibernetika*, 1966, n° 5.
60. UNGER S. H. Asynchronous sequential switching circuits. N.Y., Wiley-Interscience, 1969.
61. UNGER S. H. Flow Table Simplification — Some Usefull Aids. *IEEE Transactions on Electronic Computers*, 1965, vol. EC-14, n° 3.
62*. VARCHAVSKI V. I. (rédacteur). Automates apériodiques. Moscou, « Naouka », 1976.
63. WEINER P., SMITH E. J. Optimization of Reduced Dependencies for Synchronous Sequential Machines. *IEEE Transactions on Electronic Computers*, 1967, vol. EC-16, n° 6.
64*. YAKOUBAITIS E. A. Synthèse des automates finis asynchrones. Riga, « Zinatne », 1970.
65. YANG C. C. Closure Partition Method for Minimizing Incomplete Sequential Machines. *IEEE Transactions on Computers*, 1973, vol. C-22, n° 12.
66*. YANOV You. I. Sur les schémas logiques d'algorithmes. « Problemy kibernetiki », 1958, fasc. 1.
67*. ZAKHAROV V. N., POSPELOV D. A., KHAZATSKI V. E. Systèmes de commande: Position du problème. Etude. Réalisation. Moscou, « Energuia », 1972.
68*. ZAKREVSKI A. D. Algorithmes de synthèse de machines séquentielles. Moscou, « Naouka », 1971.
69*. ZAKREVSKI A. D., BALAKLEY L. I., ELISÉEVA H. A. et autres (réd.). Synthèse des automates asynchrones. Minsk, « Naouka i tekhnika », 1975.
70*. ZAKREVSKI A. D. Synthèse logique des circuits à plusieurs étages. Moscou, « Naouka », 1981.

INDEX ALPHABÉTIQUE DES MATIÈRES

Aléas (dans un automate) 130
— admissibles 131
— critiques 131
— non critiques 131
Alphabet d'entrée 25
— — externe 79
— — interne 79
— des états 25
— de sortie 25
— structuré 109
Application bijective 21
— biunivoque 21
— surjective 21
Automate abstrait 25
— asynchrone 35
— complètement spécifié 27
— composite 41
— déterministe 30
— fini 27
— incomplètement spécifié 27
— initialisé 30
— de *Mealy* 27
— microprogrammé 208
— de *Moore* 27
— partiel 27
— résultant 70
— structuré 109
— synchrone 35
Automates complets 111
— équivalents 36, 66
— isomorphes 84

Bascule à commande asymétrique 123
— — symétrique 122
— du type D 121
— du type RS 123
— du type T 121
Bijection 21

C-automate 41
Chaîne d'états d'un automate 51
Chemin de transition 210
Classe de compatibilité 45
— — essentielle 53
— — maximale 47
— — première 54
— d'équivalence 22
Codage des états de l'automate 130
— — — par adjacence 132
— des micro-instructions 276
Compatibilité en sortie 45
— d'états 47
Composition des automates 83
Compteur élémentaire 122
Conditions logiques 144
Conjonction déterminante 197
Connexion des automates 70
— à boucle de retour 74
— à fonction de sortie 77
— en parallèle 70
— en série 72
Correspondance binaire 17
— fonctionnelle 21
— partout définie 19
— vide 19
—, coupe droite 19
—, coupe gauche 19
—, image 19
—, image réciproque 19
Cortège 16
—, composante 16
—, couple 17
—, couple symétrique 20
—, longueur 17
—, *n*-uplet 17
—, triplet 17
—, triplet caractéristique 105
Couple de partitions 93
Couples de codes 131
— découplés 131
— liés 132
— non liés 131
Courses dans un automate 130
Cube(s) 167
0-cube 167
1-cube 167
2-cube 167
— marqué 167
—, recollement 170
—, recollement complet 170

Décomposition des automates 95, 31
— — parallèle 107
— — série 107
— des matrices 285
Dichotomie 23
Distance de *Hamming* 137
Distribution des décalages 148, 187
— — universelle 187
— — vide 187
Domaine d'arrivée (d'une correspondance) 17
— de définition 19
— de départ 17
— des valeurs 19

Ensemble 13
— de conditions logiques 144
— engendré par une classe de compatibilité 54
— fini 13
— des fonctions de transition complet 143
— — — orthogonal 143
— infini 13
— irréductible 66
— partiellement ordonné 24
— universel 15
— vide 14
—, complémentaire 16
—, élément 13
—, membre 13
—, point 13
Ensembles égaux 13
— enchaînés 56
— identiques 13
—, différence 16
—, intersection 15
—, produit 17
—, réunion 16
Equations canoniques 110
Etat impossible 45
— initial 25
— stable 34
— transitoire 38
Etats compatibles 45
— équivalents 64
— incompatibles 45
— k-distinguables 67
— k-équivalents 67
— 0-équivalents 69
Evolution d'un automate 32

Factorisation des fonctions booléennes 239
— — — vers le bas 243
— — — vers le haut 240
Fonction booléenne 159
— — marquée 159
— —, représentation géométrique 167
— de coût 140
— définie partout 21
— des entrées de l'élément de mémoire 114
— d'état final 30
— d'excitation de la mémoire 113
— — complétée 228
— particulière réduite 189
— de retour 113
— de rétroaction 113
— des sorties 25
— de transition entre opérateurs 146
— — finale 31
— des transitions 25

Formule de transition 158
— parenthétisée 160
—, recouverement 167
—, réduction 159
—, système 159

Graphe d'un automate 29
— orienté 29
Groupement régulier 52

Isomorphisme des automates 84

Matrice logique programmable (MLP) 293
— structurelle d'un réseau d'automates 106
Mémoire d'un automate 111
Méthode canonique de synthèse des automates 110
— de *Petrick* 178
Micro-instruction 142
Micro-opération 142
Microprogramme 143
Minimisation des automates 45
— — microprogrammés 218
— des schémas-graphes d'algorithmes 164
— — — par fusionnement 195
— — — par la méthode de la distribution des décalages 187
— — — — par réduction des sommets conditionnels 164
— — — — — — opératoriels 191
Mot d'entrée 43
— — applicable 43

Nœud d'un SGA 232

Opérateur primaire 172
Opération d'absorption 172
— de recollement des cubes marqués 170
— — complet 170
Ordre lexicographique 23
— partiel 23
— strict 23
— total 23
— — strict 23

Partition (d'un esemble) 22, 85
— en classes d'équivalence 22
— primaire 88
— à propriété de substitution 89
— unité 86
— zéro 86
—, couple de partitions 93
—, produit 87

—, π-image 90
—, somme 87
Prédécodeur de retour 226
Problème de décomposition d'un automate 85
— de minimisation d'un automate partiel 45
Projection première 19
— seconde 19

Réalisation d'un automate 85
— matricielle des réseaux combinationnels 258
— — triviale 262, 309
— d'un schéma-graphe 213
Recouvrement fermé 52
— minimal 52
Régistre élémentaire 123
Relation binaire 18
— diagonale 22
— d'équivalence 22
— d'ordre 21
— — partiel 22
— — strict 23
— — total 23
— — — strict 23
Réponse d'un automate 32
Réseau (d'automates) 78
— combinationnel 27
—, automate composant 79
—, automate résultant 80
—, base 79
—, fonction de connexion 79
—, fonction d'entrée 79
—, fonction de sortie 79
—, matrice structurelle 106
—, structure 79

Schéma-graphe d'algorithme 143
— — — explicité 149
— — — marqué 208
— logique d'algorithme 151
— matriciel 162
Semi-automate 77
Séquence de collections 148
— — admissible 148
Sommet conditionnel 144
— final 143
— initial 143
— opératoriel 144
Sous-ensemble 14
— impropre 14
— propre 14
Surjection 21
Symétrisation 20
Syntèse d'un automate microprogrammé 208
— — — de *Mealy* 208
— — — de *Moore* 214
— — avec des MLP 292
— — — — — combinationnelles 298
— — — — — à mémoire 320
— structurelle 109
— — avec des bascules D, T, RS 121
— — , méthode canonique 109
Système d'automates structurellement complet 110
— des transitions complet 111

Table composite 29
— de recouvrement et de fermeture 56
— de structure 223
— des transitions 28, 217
— — directe 218
— — inverse 218
— — marquée 28
— triangulaire 46
Théorème général de décomposition 95
— sur le système d'automates structurellement complet 110

Imprimé en Union Soviétique